TRAINING LARGE-SCALE VIDEO GENERATIVE ADVERSARIAL NETWORKS FOR HIGH QUALITY VIDEO SYNTHESIS

BY

XIAODAN DU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Associate Professor Svetlana Lazebnik

ABSTRACT

Video synthesis using deep learning methods is an important yet challenging task for the computer vision community. Generative Adversarial Networks have been proved effective for generating high fidelity photo-realistic images. Recently, many video synthesis models achieve high fidelity and resolution samples by carrying the success of Generative Adversarial Networks to the field of video synthesis. However, it can be challenging to train large-scale Generative Adversarial Networks as they often require enormous computing resources and a long training period. We found it necessary to put together a clear and in-depth guideline for researchers who are interested in training large-scale video Generative Adversarial Networks in the future. In this thesis, we aim to find effective and efficient ways to implement and train large-scale video Generative Adversarial Networks for high quality video generation. We evaluate different implement choices as well as training details and give quantitative analysis.

To my parents, friends and mentor for their love and support.

ACKNOWLEDGMENTS

I would like to thank Associate Professor Svetlana Lazebnik for her guidance and encouragement throughout my entire graduate study. Professor Lazebnik sparked my interest in deep learning and provided great support to me at every stage of my study as a graduate student. I wish to thank Ph.D. candidate Daniel McKee, who patiently guided and inspired me throughout this thesis work. I would also like to thank my fellow graduate student and good friend, Jugat Lamba. Your kindness and passion gave me courage to overcome frustrations and difficulties along the way. I wish to express my appreciation toward the Department of Computer Science for admitting me to this program. I would also like to thank Amazon Web Services's support which made this work possible.

Most importantly, I want to thank my beloved parents Jie and Jianli for their support and encouragement. I owe a special thanks to my girlfriend Yinong for her understanding and support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

One of the most important characteristics of human intelligence is the ability to create based on past experience. Recently, the increasing computing power available to the public combined with deep neural networks has revolutionized Artificial Intelligence (AI). Deep learning models have achieved or even exceeded human level performance on *discriminative* tasks such as image classification [1] and object detection [2]. Creation, or *generative* tasks, on the other hand, is considered more difficult prior to the introduction of Generative Adversarial Networks (GANs) [3]. GANs bring a lot of advancement in visual data synthesis. With the help of GANs, deep neural networks are able to generate high resolution, high fidelity images of human faces [4, 5], realistic photographs [6] and artistic pictures [7, 8].

Video synthesis is more challenging than image synthesis as it deals with larger and more complicated training data and requires frames in a single clip to be temporally coherent [9]. Encouraged by the massive achievement of GANs in the field of image synthesis, researchers begin to apply adversarial learning methods on video synthesis tasks and see a significant boost in performance [9, 10, 11, 12]. Nevertheless, GANs, especially large-scale ones, still suffer from some common issues including instability during training, overfitting, restricted scalability for complex datasets and large search space for hyperparameter optimization. All these issues create additional burdens for individual researchers without a plethora of computational resources to train large-scale video GANs. Even though recent publications provide various new training models, very few of them provide detailed guidelines on how to implement and train these models. We take the state-of-the-art video generative model, DVDGAN [12], and a large-scale video dataset, Kinetics-400 [13], as an example in order to compare the performances of different design choices and training strategies. Furthermore, we show strategies to avoid problems such as overfitting and instability.

1.1 GENERATIVE ADVERSARIAL NETWORKS

Generative tasks are considered a lot harder than discriminative tasks for deep neural networks partly because it is hard to do maximum likelihood estimation for generation [3]. For example, if two generative models are asked to create paintings mimicking the artistic style of Pablo Picasso, it will be hard to judge which model does a better job as which painting is "more similar to Picasso's art style" is hard to quantify. Generative Adversarial Networks solve such problems by jointly training two separate deep neural networks: a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$ [3]. During training time, $\mathcal{G}$'s goal is to generate samples

that are "realistic" enough to deceive $\mathcal{D}$ while $\mathcal{D}$'s goal is to distinguish "fake" samples from "real" ones. This adversarial game can be formulated as a two-player minimax game given below:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{F}(\mathcal{D}, \mathcal{G}). \tag{1.1}$$

Given latent vector $\mathbf{z} \in \mathbb{R}^d$ from a distribution P_z, the goal function is designed so that $\mathcal{G}$ maps $\mathbf{z}$ to $\widetilde{\mathbf{x}} \in \mathbb{R}^m$, where m is the dimension of the training samples. On the other hand, $\mathcal{D}$ maps sample $\mathbf{x} \in \mathbb{R}^m$ to a scalar in range of $\{0, 1\}$ [14]. $\mathcal{D}$'s goal is to classify input samples as "fake" or "real" and maximize the loss function. The overall loss function introduced in [3] is then given by:

$$\mathcal{F}(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{\mathbf{x} \sim P_{data}(\mathbf{x})}[\log \mathcal{D}(x)] + \mathbb{E}_{\mathbf{z} \sim P_z(\mathbf{z})}[\log(1 - \mathcal{D}(\mathcal{G}(z)))], \tag{1.2}$$

where $\mathbf{x}$ is the data sample, P_{data} is the probability distribution over data samples, $\mathbf{z}$ is the latent vector and P_z is the probability distribution over latent vector. In practice, $\mathcal{G}$ and $\mathcal{D}$ are updated alternatively to prevent $\mathcal{D}$ from overfitting [3].

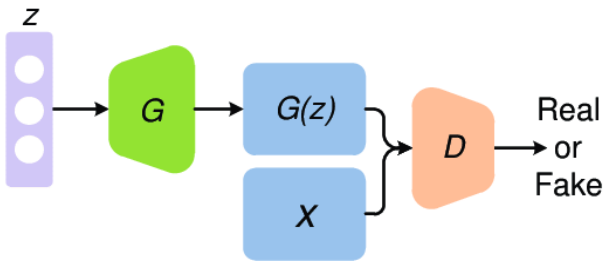


Figure 1.1: Typical GANs Architecture: Image taken from [15].

Since the introduction of GANs, many improved models have been proposed. DCGAN's [16] architecture has influenced a lot of GANs after it. DCGAN uses deep convolutional neural networks for both the generator and the discriminator. DCGAN also shows that Batch Normalization [17] is useful for mitigating mode collapse. LSGAN [18] adopts the least squares loss function for $\mathcal{D}$ to replace the sigmoid cross entropy loss function in previous GANs for better stability and image quality. WGAN proposed in [19] adopts a new loss function using Earth Mover (EM) distance to achieve a better stability and provide a meaningful loss metric that correlates to the quality of output samples. SAGAN [20] applies two very important techniques, self-attention mechanism [21] and spectral normalization [22], to convolutional GANs and achieves better results on the ImageNet dataset [23] than previous models. Both techniques are later inherited by BigGAN [6] and DVDGAN [12].

1.2 VIDEO SYNTHESIS WITH DEEP NEURAL NETWORKS

Different deep learning models have been proposed to solve different video synthesis problems. Most of the video synthesis tasks can be broadly categorized by how much conditioning signal is available to the model. On one end of the spectrum is *unconditional* video synthesis models. On the other end is *strongly-conditioned* video synthesis models. *Class-conditional* video synthesis models fall in the middle of the spectrum.

Unconditional video synthesis. Similar to unconditional image GANs, some video GANs train a generator $\mathcal{G}$ with only random noise vectors. Since the training data is not labelled in any form, the samples generated by $\mathcal{G}$ follow the distribution of the entire training set. VGAN [9] is among the first models to extend the success of GANs to the video domain. Instead of one 3D generator, TGAN [10] uses a *temporal generator* $\mathcal{G}_0$ to yield latent vectors that will then be "decoded" by the *image generator* $\mathcal{G}_1$ into temporally coherent video samples. MoCoGAN [11] assumes a video clip consists two parts: a content part and a motion part. For each generated sample with T frames, T motion codes sampled by a recurrent neural network (RNN) and a single content codes are combined and sent into $\mathcal{G}$. TGANv2 [24] subsamples feature maps and adaptively reduces batch size in the generator to reduce memory and computational costs. In general, lack of conditioning signal restricts unconditional models' ability to generate long and high-resolution videos [25].

Class-conditional video synthesis. Most class-conditional video synthesis models are conditional video GANs that embed class information into a latent vector. In [10], a class-conditional version of TGAN, CTGAN is proposed as an extension of TGAN. CTGAN tranforms a class label l into an one-hot vector v_l and concatenates it with noise vector z_0 to form the new latent code for generation. DVDGAN [12], which we use in this work for experiments, concatenates learned linear embedding of class with sampled random noise. DVDGAN also uses projection-based $\mathcal{D}$ and class-conditional Batch Normalization in $\mathcal{G}$ to impose class conditioning.

Strongly-conditioned video synthesis. Sometimes conditioning signal can be more fine-grained than class label, such as pixel-level masks [25] or pose stick figures [26]. Many deep learning models [26, 27, 28] have been proposed for a class of problems called "content transfer". Some strongly-conditioned video synthesis problems require per-frame segmentation masks [25]. Video inpainting [29, 30] is another type of strongly-conditioned problem that is worth mentioning. Because the conditioning signal and problem setup for

different strongly-conditioned video synthesis tasks can be very different, normally it is difficult to apply strongly-conditioned models to different applications [25].

CHAPTER 2: RELATED WORK

2.1 GANS FOR IMAGE SYNTHESIS

A lot of effective techniques originally developed for image GANs are adopted by GANs for video. DVDGAN [12], the model we aim to implement in this work, borrows ideas and methods proposed by *Self-Attention Generative Adversarial Networks* (SAGAN) [20] and *Large Scale GAN Training for High Fidelity Natural Image Synthesis* (BigGAN) [6]. We find a brief introduction of the two models may be helpful.

2.1.1 SAGAN

The separable-attention module introduced in [12] is based on the self-attention module proposed in [20] for image generation tasks. A common shortcoming of GANs is that convolutional layer often struggles to model long-range dependencies across widely separated regions in images. For example, an image GAN may generate realistic fur texture of a dog but fail to model its four legs well. [20] argues the small size convolution kernels (usually 3×3 or 5×5) used by recent convolutional neural networks are good at catching local and low-level features but are not sufficient to represent long-range spatial structures. Increasing the size of the kernels is also not an option as it dramatically increases the number of parameters.

SAGAN solves the dilemma by applying *self-attention*, a mechanism previously used by natural language processing models [31, 32, 21] to an image GAN. Self-attention obtains knowledge of global structural patterns and long-range dependencies by directly computing how every "pixel" in the feature maps is related all other "pixels" from the same layer. The formation of the self-attention module in SAGAN is adapted from the *Non-local Neural Networks* [33] and illustrated in Figure 2.1. The features obtained from the previous layer $\mathbf{x} \in \mathbb{R}^{C \times N}$ are passed into two 1×1 convolutions f and g separately, where C is the number of channels and N is the multiplication of the width and height of the feature maps. The "attention" is obtained by taking the matrix multiplication of $f(\mathbf{x})$ and $g(\mathbf{x})$ and applying softmax operation along each row. Formally, the amount of impact that the i^{th} location has on the generation of the j^{th} location, $\beta_{j,i}$, is given by:

$$\beta_{j,i} = \frac{exp(s_{ij})}{\Sigma_{i=1}^{N} exp(s_{ij})}, \tag{2.1}$$

where $s_{ij} = f(x_i)^\top g(x_j)$. The output of the self-attention layer $\mathbf{o}$ is thus given by:

$$\mathbf{o}_j = v\left(\Sigma_{i=1}^{N}\beta_{j,i}h(\mathbf{x}_i)\right), \tag{2.2}$$

where h and v are both 1×1 convolutions, too.

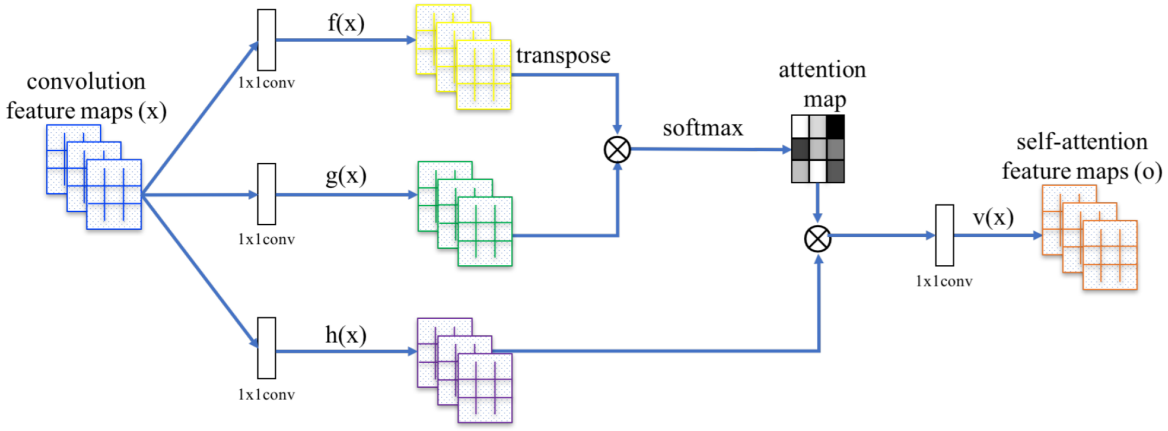


Figure 2.1: A visualization of the self-attention module. The $\otimes$ represents tensor multiplication: Image taken from [20]

The self-attention module, combined with other useful techniques adopted by SAGAN, including the spectral normalization [22] and the hinge version of the adversarial loss [34, 35, 36], has been proven effective in experiments. SAGAN increases the state-of-the-art IS for class-conditional image generation on ImageNet [23] from 36.8 to 52.52. It also achieves a lower FID (18.65) than the previous state-of-the-art image GANs [20].

2.1.2 BigGAN

BigGAN [6] is a class-conditional image synthesis GAN. It achieves a FID score as low as 8.7 for 128×128 video, less than a half of the previous state-of-the-art result of 18.65 [20]. DVDGAN is largely based on BigGAN's architecture.

BigGAN adopts the hinge formulation of GAN loss that is also used by SAGAN [20] and is inherited by DVDGAN [12]:

$$\begin{aligned}
\mathcal{D} &: \min_{\mathcal{D}} \mathbb{E}_{\mathbf{x}\sim P_{data}(\mathbf{x})}[\min\{0, 1-\mathcal{D}(x)\}] + \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})}[\min\{0, 1+\mathcal{D}(\mathcal{G}(z))\}], \\
\mathcal{G} &: \max_{\mathcal{G}} \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})}[\mathcal{D}(\mathcal{G}(z))].
\end{aligned} \tag{2.3}$$

One of the main contributions of BigGAN is pointing out that scaling up the model and batch size significantly improves performance of GANs. While SAGAN [20], the previous state-of-the-art model, uses a batch size of 256, BigGAN increases it by a factor of 8. That modification alone increases Inception Score (IS) by 46% [6]. Increasing the number of channels for the networks from 64 to 96 is also proven to be beneficial for achieving higher IS score.

Previously, class-conditional GANs such as [20] require separate class embeddings c for different layers in $\mathcal{G}$. Having separate embeddings results in heavy computation and memory burden. BigGAN solves the problem by linearly projecting a shared embedding to different layers separately. [6] reports a 37% faster training speed with shared embedding technique. Furthermore, instead of passing noise vector z to the initial layer of $\mathcal{G}$ only, BigGAN feeds the latent vector z directly into multiple layers of $\mathcal{G}$. z is cut into multiple chunks, each concatenated to c and projected to a different class-conditional BatchNorm layer of $\mathcal{G}$. Figure 2.2 is a simplified structure layout for $\mathcal{G}$ of BigGAN. In addition, BigGAN also confirms the effectiveness of design choices such as the Truncation Trick and Orthogonal Regularization.

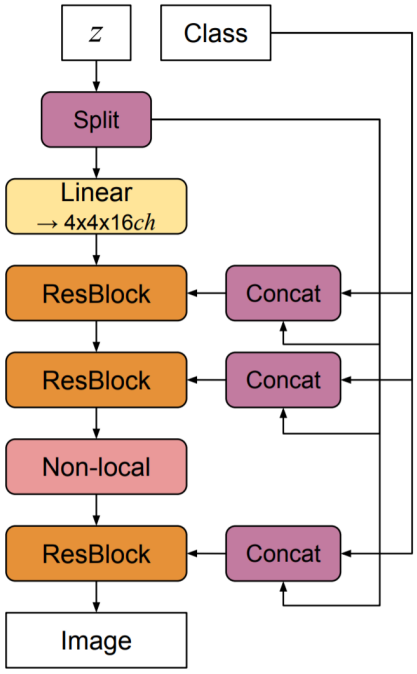


Figure 2.2: A simplified architecture layout for BigGAN's $\mathcal{G}$: Image taken from [6]

The low-level building blocks of the BigGAN architecture is similar to those in the ResNet [37] GAN architecture [20], which are shown in Figure 2.3.

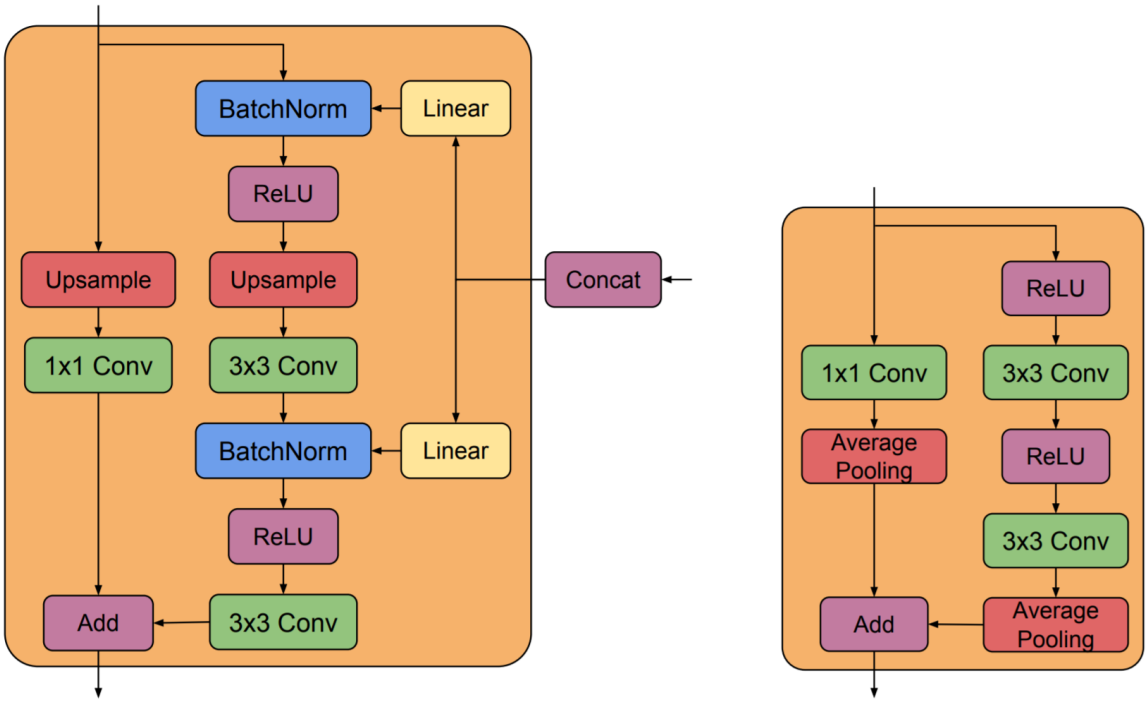


Figure 2.3: Typical residual block in BigGAN's $\mathcal{G}$ (left) and two $\mathcal{D}$s (right). Image taken from [6].

2.2 GANS FOR FIXED-LENGTH VIDEO SYNTHESIS

GANs for video broadly fall into two categories: models generating fixed-length videos and models that allow various-length video synthesis. The former does 3-D (width, height and time) generation of videos as blocks. Limited by their architectures, such models can only generate videos with the same length. VGAN [9] and TGAN [10] are typical examples of this category.

2.2.1 VGAN

VGAN [9] is among the first efforts to generate temporally coherent video samples with GAN architecture. Video signals are decomposed into two separate streams: a static "background" stream and a moving "foreground" stream. Leveraging large amounts of unlabelled video, Vondrick *et al.* proposes VGAN, a GAN based unconditional generative model that learns to model realistic foreground motions and a static background. The visualization of VGAN's architecture is in Figure 2.4. The input latent code to the model is a low-dimensional noise vector z. The latent code is up-sampled by two independent pathways

f and b. The foreground pathway f uses fractionally-strided spatio-temporal convolutional layers to map $z \in \mathbb{R}^d$ to a spatio-temporal cuboid $f(z) \in \mathbb{R}^{h \times w \times t \times c}$, where c is number of channels (3 in this case), h is the height, w is the width and t is the number of frames in a sample. Because the background is stationary throughout the entire video, the background pathway b contains only standard 2-D convolutional layers. z is up-sampled and mapped to a 2-D image $b(z) \in \mathbb{R}^{h \times w \times c}$. A network m that shares same parameters with f but the last layer is created to produce a $h \times w \times t \times 1$ mask tensor $m(z)$. $m(z)$ is further constrained by a Sigmoid function to ensure $0 \leq m(z) \leq 1$. $m(z)$ is used to determine whether to use $f(z)$ or $b(z)$ at a specific pixel and frame. With the help of $m(z)$, a generated video sample $\mathcal{G}(z)$ is synthesized following the formula:

$$\mathcal{G}(z) = m(z) \odot f(z) + (1 - m(z)) \odot b(z), \tag{2.4}$$

where $\odot$ is element-wise multiplication.

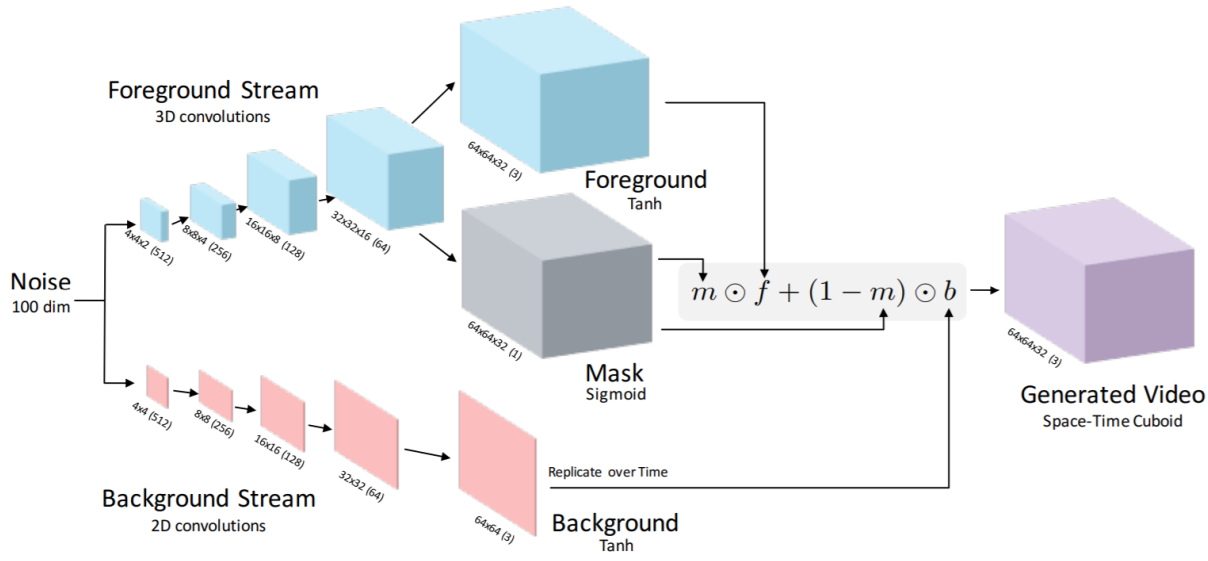


Figure 2.4: A simplified architecture layout for VGAN: Image taken from [9]

As the first attempt to carry the success of GAN in the image domain to the video domain, VGAN has demonstrated a feasible way to learn motion dynamics. Experiments have also suggested VGAN is able to learn features for human action classification without any supervision. However, the explicit separation of the foreground and the background makes VGAN incapable to learn more complicated videos. The assumption of a stationary background greatly restricts the usefulness of VGAN. The relatively simple and primitive network structures and loss functions prevent the model to produce videos of higher visual quality. More importantly, VGAN's architecture does not allow for generating videos with

arbitrary lengths.

2.2.2 TGAN

Unlike VGAN, TGAN [10] contains no 3-D convolutional layers in its generators. Instead, TGAN generates frames and temporal dynamics separately. Given a noise vector z_0, the temporal generator $\mathcal{G}_0$ maps it to a series of latent codes, each represents a latent point in the image domain. The image generator $\mathcal{G}_1$ takes in both z_0 and generated latent codes to produce video frames. TGAN increases the state-of-the-art IS of UCF101 from 8.31 to 15.83. TGAN's architecture is in Figure 2.5.

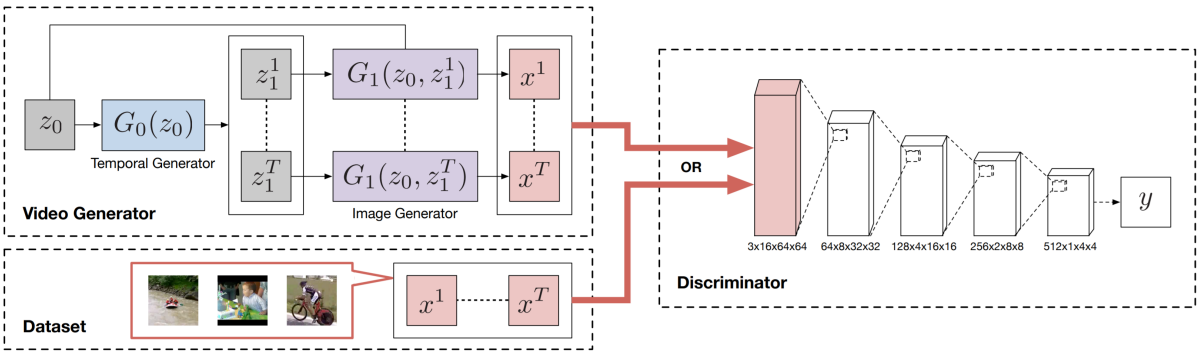


Figure 2.5: A simplified architecture layout for TGAN: Image taken from [10]

2.3 GANS FOR VARIABLE-LENGTH VIDEO SYNTHESIS

The other type of video GANs sequentially generate video frames by a recurrent neural network (RNN). The RNN allows the model to generate arbitrary number of frames. MoCoGAN [11] and DVDGAN [12] belong to this category.

2.3.1 MoCoGAN

Both VGAN [9] and TGAN [10] assume a video sample as a point in a latent space. Their architectures are designed to convert a point in the latent space to a video clip. Tulyakov *et al.* [11] argue that such an assumption unnecessarily makes the problem more complex. First of all, videos with the same action but different speed are considered different points in this approach. More importantly, this approach disregards the fact that videos have different lengths. Alternatively, Tulyakov *et al.* assume a video is made by traversing

multiple points in a latent space of *images*. Under this assumption, videos of different speed and lengths correspond to trajectories in the latent space traversed with different speed and lengths. The new approach makes it possible to further decompose the latent space of images into two subspaces: the *content* subspace and the *motion* subspace. Based on the assumption, an unconditional video GAN, the Motion and Content decomposed Generative Adversarial Network (MoCoGAN) is proposed. Instead of directly mapping a latent vector to a video, MoCoGAN generates temporally coherent frames sequentially. The latent code for each frame consists a "content" code which is randomly sampled from Gaussian distribution and is fixed throughout the entire video clip and a "motion" code which is generated by a learned recurrent neural network. Fixing the "content" representation while having changing "motion" representation encourages the model to differentiate the two subspaces and generate realistic videos which contain consistent objects and coherent motions.

The MoCoGAN consists of an one-layer GRU network R_M, an image generator network $\mathcal{G}_I$, an image discriminator network $\mathcal{D}_I$ and a video discriminator network $\mathcal{D}_V$. During training, the "content" code Z_C is sampled once and fixed for the entire video. A series of randomly sampled vectors $[\epsilon^{(1)}, \ldots, \epsilon^{(K)}]$ are input into the R_M and encoded to K "motion" codes $[Z_M^{(1)}, \ldots, Z_M^{(K)}]$ one at a step. The image generator $\mathcal{G}_I$ maps concatenated Z_C and $Z_M^{(k)}$ to the kth frame $\widetilde{\mathbf{x}}^{(k)}$. $\mathcal{D}_I$ is a standard 2-D CNN architecture and only assesses the spatial quality of each single frame. $\mathcal{D}_V$ contains spatio-temporal CNN layers so that it can take in a fixed-length video clip and tell if it is synthetic or authentic. A visual illustration of MoCoGAN's architecture can be found in Figure 2.6.

Although MoCoGAN has surpassed its predecessors such as VGAN and TGAN in terms of quantitative evaluation metrics on several datasets, we find it important to point out the following problems:

- The decomposition of visual signals in a video to a "content" component and a "motion" component oversimplifies the problem by assuming all videos must have a fixed and well-defined object and a still background. This assumption may help the MoCoGAN to do well on highly restrained datasets like MUG Facial Expression Database [38] and Weizmann Action database [39]. When it comes to more complex and diverse datasets like UCF101, MoCoGAN has difficulty presenting multiple objects, deformable objects or camera panning in output samples. According to our experiments, MoCoGAN may achieve a slightly better Inception Score than TGAN and VGAN on UCF101 [40], the visual quality of the synthetic samples is still very bad.

- Even for those datasets that MoCoGAN performs better on, the improvements could

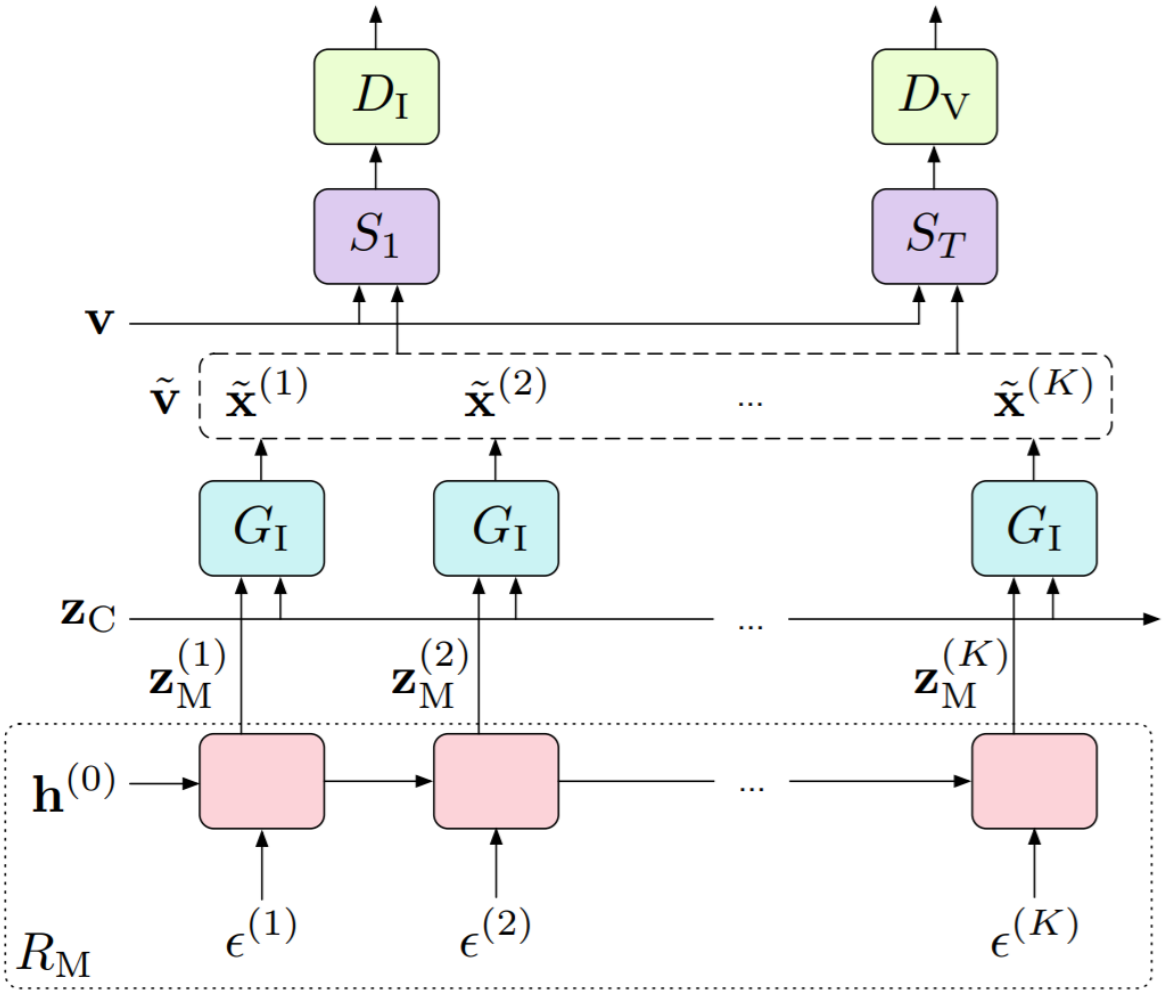


Figure 2.6: A simplified architecture layout for MoCoGAN: Image taken from [11]

be the result of GAN memorization. Datasets used in the experiments are very small. Except UCF101, the largest dataset used in training contains only 4,500 clips, nearly 70 times smaller than Kinetics-400 [11, 13].

2.3.2 DVDGAN

The success of BigGAN has been extended to the video domain. Dual Video Discriminator GAN (DVDGAN) [12], a class-conditional GAN architecture that is largely based on BigGAN is proposed for video synthesis and video prediction. We focus on the video synthesis part in our work.

Most video GAN models [9, 10, 11] prior to DVDGAN mainly deal with small and simple video datasets including the Moving MNIST dataset [41] and the Weizmann Action database

[39]. By leveraging the scalability of BigGAN's architecture, DVDGAN is able to produce high fidelity 256 × 256 videos with 48 frames and achieve state-of-the-art results on both UCF101 [40] and Kinetics-600 [42] datasets. Because of the strong learning ability of its neural networks, DVDGAN does not explicitly separate foreground from background as in [11] or generate new frames via flow warping and the hallucination network as in [25]. DVDGAN contains two discriminators: a *spatial discriminator* $\mathcal{D}_S$ and a *temporal discriminator* $\mathcal{D}_T$. The main innovations of DVDGAN are the use of Convolutional GRU and the introduction of *Separable Attention* module, a spatio-temporal extension of the self-attention module proposed by [20] for image synthesis. For experiments generating less than 48 frames, [12] uses the Multiplicative Convolutional GRU. We use the standard Convolutional GRU in our training to reduce memory and computational costs. Most low-level architectural choices confirmed effective by BigGAN are inherited by DVDGAN.

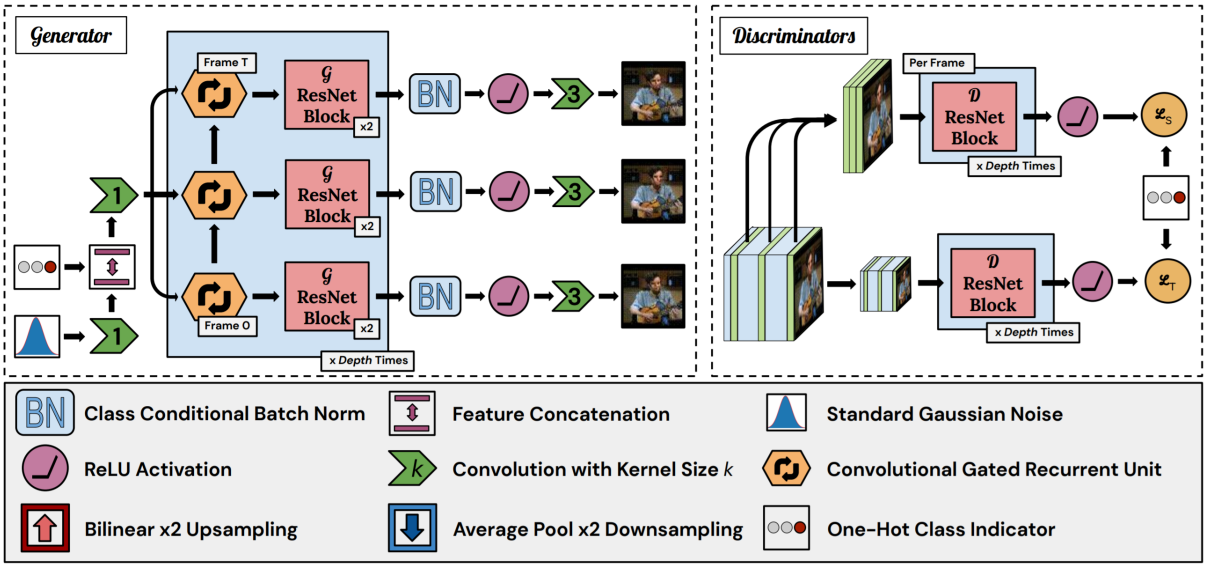


Figure 2.7: DVDGAN architecture. Left: $\mathcal{G}$; right: $\mathcal{D}_S$ and $\mathcal{D}_T$. Image taken from [12].

2.4 DATASETS

2.4.1 Kinetics datasets

DeepMind's *Kinetics human action video dataset* (Kinetics-400) [13] was first released in 2017, containing 400 classes, with at least 400 video clips per class. The dataset focuses on human actions. Each class contains 400 to 1150 video clips, each around 10 seconds. Certain procedures are taken to make sure that each clip is from a unique video and cannot

be assigned to more than one class. There are a total of 306,245 videos in Kinetics-400 dataset. Kinetics-400 is later expanded to 600 classes with at least 600 video clips per class, thus called Kinetics-600 [42]. The overall number of videos increases to 495,547. A small portion of classes (32 out of 400) from Kinetics-400 are changed in Kinetics-600. DVDGAN uses Kinetics-600 for training and evaluation [12]. We use Kinetics-400 instead because Kinetics-600 is not publicly available anymore when we start the work. Furthermore, since we use PyTorch's public available *ResNet (2+1)D* model trained on Kinetics-400 to report Inception Score (IS) [43] and Fréchet Inception Distance (FID) [44], training the model on the same dataset makes better sense.

2.4.2 UCF101

UCF101 [40] is widely used for training generative models before the introduction of Kinetics dataset. UCF101, collected from YouTube, is originally designed for action recognition tasks. It is an extension of its predecessor UCF50 which has only 50 classes. UCF101 contains 13,320 video clips at 320 $\times$ 240 resolution and 25FPS frame rate. Besides its smaller size, UCF101's biggest problem compared to Kinetics datasets is the lack of intra-class diversity. Each class of UCF101 is consist of 25 *clip groups*. Each clip group contains 4 to 7 video samples. In fact, video samples within a single clip group are just different clips from the same video. As a result, GANs trained on UCF101 dataset tend to overfit more easily. In section 4 we use UCF101 to show how DVDGAN memorize training set that is not complex and diverse enough.

Dataset	**Classes**	**Clips per Class**	**Total Clips**	**Mean Clip Length**
UCF101	101	min 101	13,320	7.21 sec
Kinetics-400	400	min 400	306,245	$\sim$ 10 sec
Kinetics-600	600	min 600	495,547	$\sim$ 10 sec

Table 2.1: Video dataset statistics. Data taken from [40, 13, 42].

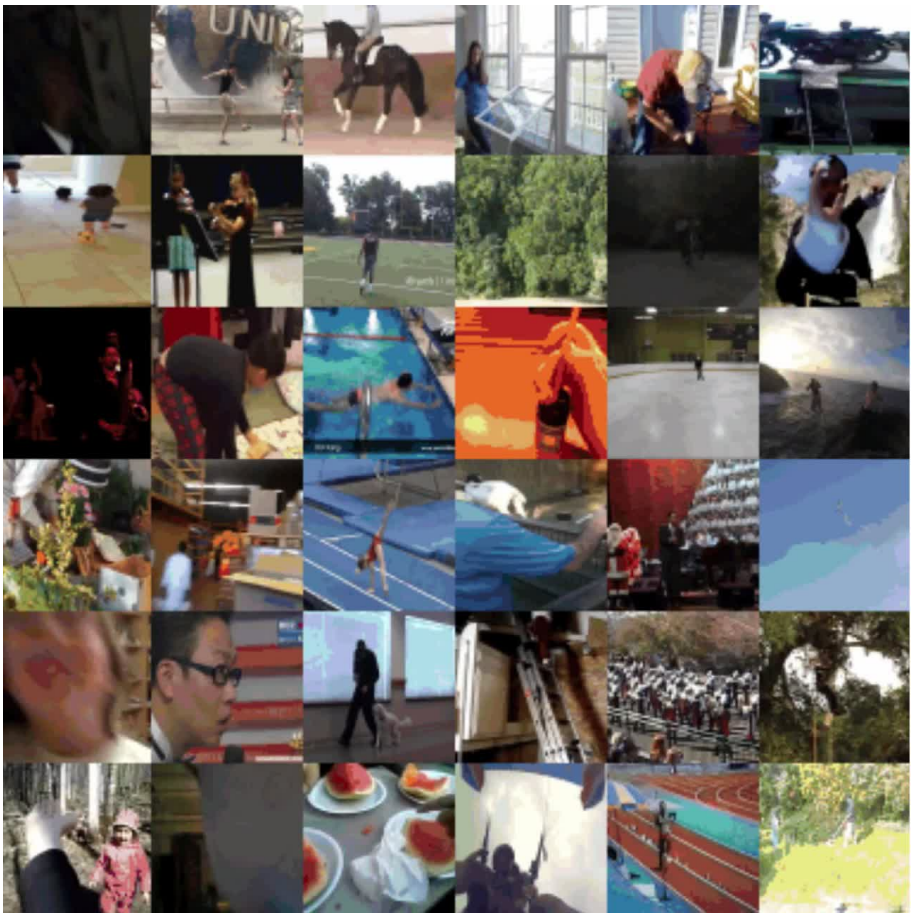


Figure 2.8: Randomly selected Kinetics-400 training samples.

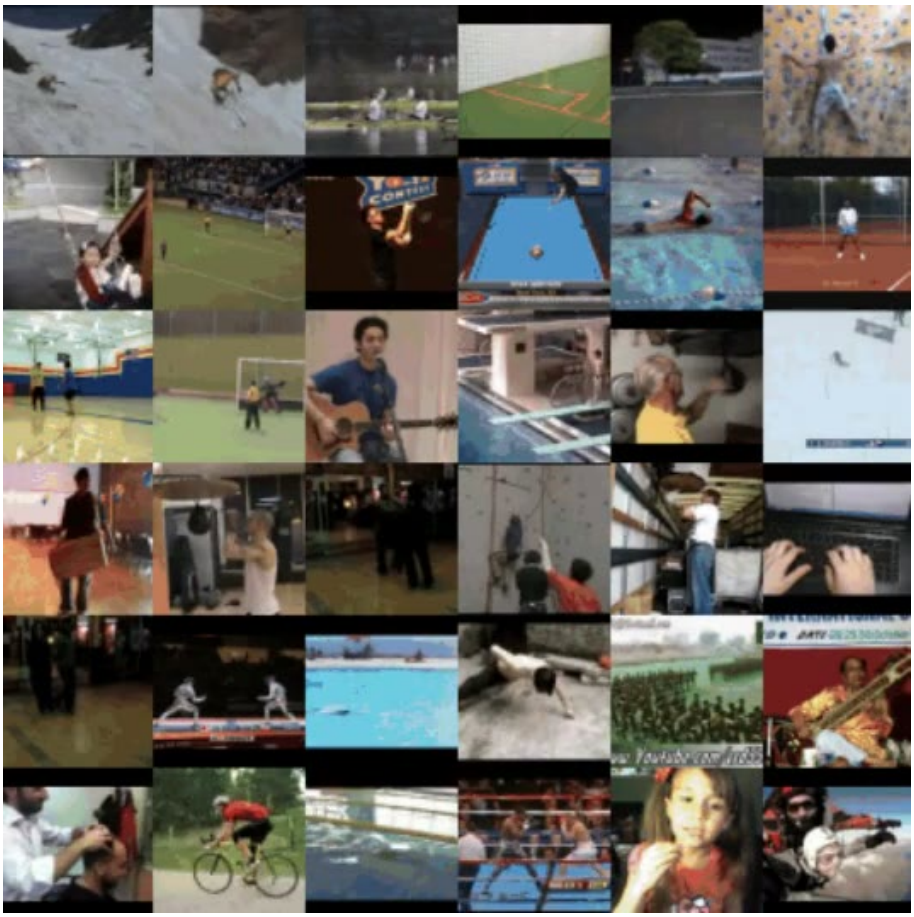


Figure 2.9: Randomly selected UCF101 training samples. Note the first and the second clips from left in the top row are actually from the same clip group (video).

CHAPTER 3: APPROACH

In this section we provide a detailed explanation of the architecture of DVDGAN and our implementation choices. Our goal is to provide a video generation model implementation similar to DVDGAN using less resources and explore implementation choices for this model.

3.1 CONVOLUTIONAL GRU

The randomly sampled latent code z and linearly embedded class label $e(y)$, which are both 128-dimensional vectors, are first concatenated and then input into a convolutional GRU network [45]. [12] uses multiplicative convolutional GRU for videos shorter than 48 frames. We use a standard single-layer convolutional GRU, whose update rule is given by:

$$\begin{aligned} r &= \sigma(W_r \star_3 [h_{t-1}; x_t] + b_r) \\ u &= \sigma(W_u \star_3 [h_{t-1}; x_t] + b_u) \\ c &= \rho(W_c \star_3 [x_t; r \odot h_{t-1}] + b_c) \\ h_t &= u \odot h_{t-1} + (1 - u) \odot c, \end{aligned} \tag{3.1}$$

where σ is the elementwise sigmoid operation, ρ is the ReLU functions, $\star_n$ is 2-D convolution with $n \times n$ kernel, $\odot$ is elementwise multiplication and square bracket represents concatenation. Note the same z is used as input to all timesteps in the convoluitonal GRU. Input vector $[z; e(y)] \in \mathbb{R}^{256}$ is mapped to a $T \times ch_0 \times 4 \times 4$ tensor, where T is number of frames per sample (in our case, 12) and ch_0 is the number of input channels of the initial layer in $\mathcal{G}$ (in our case, 512). Comparing with previous works with the regular GRU [11], DVDGAN provides a stronger spatial prior to $\mathcal{G}$.

3.2 GENERATOR

A non-causal self-attention block is applied on the output of the convolutional GRU before it is sent into the generator. The self-attention block is applied directly across the width, height and time dimension and is not separable. DVDGAN's generator $\mathcal{G}$ takes in the entire output from the self-attention block and generates frames in parallel. DVDGAN's $\mathcal{G}$ is very similar to that of BigGAN [6], containing 2-D convolutional layers and the Batch Normalization layers. Spectral Normalization [22] is applied to all weight layers. $\mathcal{G}$'s residual block

is identical to what is shown in Figure 2.3. A layout of the generator's overall architecture we use for the experiments can be found in Table 3.1. Because $\mathcal{G}$ does not have any 3-D convolutional layers, the time axis of the latent code tensor needs to be folded into the batch axis before the forward pass. Each frame is produced independently. Since dynamic information of the sample is already provided by the convolutional GRU, it is important to keep the networks from "cheating" by communicating dynamic knowledge through the Batch Norm layers. Therefore, the time axis is unfolded from the batch axis and folded into the channel axis every time before passed into the Batch Norm layers.

2-D latent code $\widetilde{\mathbf{z}} \in \mathbb{R}^{B \times 12 \times ch * 8 \times 4 \times 4}$
ResBlock up $[B * 12, ch * 8, 4, 4] \rightarrow [B * 12, ch * 8, 8, 8]$
ResBlock up $[B * 12, ch * 8, 8, 8] \rightarrow [B * 12, ch * 8, 16, 16]$
ResBlock up $[B * 12, ch * 8, 16, 16] \rightarrow [B * 12, ch * 4, 32, 32]$
ResBlock up $[B * 12, ch * 4, 32, 32] \rightarrow [B * 12, ch * 2, 64, 64]$
ResBlock $[B * 12, ch * 2, 64, 64] \rightarrow [B * 12, ch, 64, 64]$
BN, ReLU, 3×3 Conv $[B * 12, ch, 64, 64] \rightarrow [B, 12, 3, 64, 64]$
Tanh

Table 3.1: DVDGAN $\mathcal{G}$ architecture for 64×64, 12 frames videos. B represents the batch size and ch represents the channel number multiplier, which is 64 for our model.

3.3 DISCRIMINATORS

DVDGAN contains two discriminators: a spatial discriminator $\mathcal{D}_S$ that only inspects spatial content and structure of individual frames; and a temporal discriminator $\mathcal{D}_T$ that critiques the entire video as a whole. We perform Spectral Normalization on all weight layers for both discriminators.

Considering memory and computational costs, k full-resolution frames (we use $k = 8$, same as [12]) will be randomly sampled from each video clip and processed by $\mathcal{D}_S$. The k axis is folded in the batch axis before forward pass. The rest of the structure is almost identical to BiGGAN's discriminator. See Table 3.2 for the complete architecture for $\mathcal{D}_S$.

The temporal discriminator $\mathcal{D}_T$ contains two 3-D residual blocks followed by two 2-D residual blocks. To cut down computational costs, video samples are downsampled by a 2×2 average pooling function before processed by $\mathcal{D}_T$. Note the time dimension is not reduced by the average pooling operation. The time dimension is folded into the batch dimension during the transition from 3-D blocks to 2-D blocks. The low-level architecture of $\mathcal{D}_T$ is similar to that of $\mathcal{D}_S$. See Table 3.3 for the complete architecture for $\mathcal{D}_T$.

RGB video frames $\widetilde{\mathbf{x}} \in \mathbb{R}^{B\times 8\times 3\times 64\times 64}$
ResBlock down $[B*8, 3, 64, 64] \rightarrow [B*8, ch*2, 32, 32]$
ResBlock down $[B*8, ch*2, 32, 32] \rightarrow [B*8, ch*4, 16, 16]$
ResBlock down $[B*8, ch*4, 16, 16] \rightarrow [B*8, ch*8, 8, 8]$
ResBlock down $[B*8, ch*8, 8, 8] \rightarrow [B*8, ch*16, 4, 4]$
ResBlock $[B*8, ch*16, 4, 4] \rightarrow [B*8, ch*16, 4, 4]$
ReLU, Global sum pooling
$e(y) \cdot \mathbf{h} + (linear \rightarrow 1)$

Table 3.2: DVDGAN $\mathcal{D}_S$ architecture for 64×64, 12 frames videos. B represents the batch size and ch represents the channel number multiplier, which is 64 for our model. $\mathbf{h}$ represents features right after the global sum pooling.

RGB downsampled videos $\mathbf{x}^* \in \mathbb{R}^{B\times 12\times 3\times 32\times 32}$
3-D ResBlock down $[B, 12, 3, 32, 32] \rightarrow [B, 6, ch, 16, 16]$
3-D ResBlock down $[B, 6, ch, 16, 16] \rightarrow [B, 3, ch*2, 8, 8]$
ResBlock down $[B*3, ch*2, 8, 8] \rightarrow [B*3, ch*4, 4, 4]$
ResBlock $[B*3, ch*4, 4, 4] \rightarrow [B*3, ch*8, 4, 4]$
ReLU, Global sum pooling
$e(y) \cdot \mathbf{h} + (linear \rightarrow 1)$

Table 3.3: DVDGAN $\mathcal{D}_T$ architecture for 64×64, 12 frames videos. B represents the batch size and ch represents the channel number multiplier, which is 64 for our model. $\mathbf{h}$ represents features right after the global sum pooling.

3.4 SEPARABLE SELF-ATTENTION

Applying a self-attention block that attends to all dimensions to large video features is not feasible since the additional time axis increases the size of the attention matrix from $\mathcal{O}((HW)^2)$ to $\mathcal{O}((HWT)^2)$. *Separable Attention* is introduced in [12] to reduce memory and computational costs of the self-attention module. Separable attention decomposes the original self-attention layer into three separate attention layers each attending only one of the time, width and height axis. The memory cost is reduced to max { $\mathcal{O}(H^2WT)$, $\mathcal{O}(HW^2T)$, $\mathcal{O}(HWT^2)$ }. We only use separable attention layer in $\mathcal{G}$, at resolution 32×32. The Python pseudocode in Figure 3.1 is given in [12] and perfectly explains the algorithm of separable attention. In the pseudocode, q, k, v and x are the query matrix, key matrix, value matrix and input feature tensor respectively.

```
def self_attention(x, q, k, v):
  xq, xk, xv = np.matmul(x, q), np.matmul(x, k), np.matmul(x, v)
  qv_correlations = np.matmul(xq, np.transpose(xk))
  return np.matmul(np.softmax(qv_correlations, axis=−1), xv)

def separable_attention(x, q1, k1, v1, q2, k2, v2, q3, k3, v3):
  b, h, w, t, c = x.shape
  # Apply attention over time.
  x = np.reshape(x, [b∗h∗w, t, c])
  x = self_attention(x, q1, k1, v1)
  # Apply attention over height.
  x = np.reshape(x, [b∗w∗t, h, c])
  x = self_attention(x, q2, k2, v2)
  # Apply attention over width.
  x = np.reshape(x, [b∗h∗t, w, c])
  x = self_attention(x, q3, k3, v3)
  return x
```

Figure 3.1: The Python pseudocode of separable attention: Image taken from [12]

CHAPTER 4: EXPERIMENTS

In this section we describe how we implement DVDGAN and discuss investigations into possible failure cases during training. All experiments are conducted on Kinetics-400 dataset unless mentioned otherwise. Every model is trained on Amazon Web Services (AWS) [46] using a single p3.16xlarge instance with 8 NVIDIA Tesla V100 GPUs. Our DVDGAN model is implemented using the PyTorch deep learning framework [47].

4.1 DATA LOADING AND PREPROCESSING

Datasets with the size of UCF101 [40] or smaller can fit on a single NVIDIA V100 GPU, which has a memory of roughly 16GiB. Larger datasets like Kinetics-400 cannot be loaded entirely on the GPU prior to the training. As a result, training data needs to be loaded on the fly. A trade-off needs to be made between I/O time and file size. Our experiments suggest the best solution is storing each frame of a video clip as a JPEG image.

We sample video frames with a stride of 2, same as [12]. Limited by computing resource available, we generate videos at 64×64 resolution and length of 12 frames. According to our experiments, different interpolation modes result in very different visual effect. Figure 4.1 shows resized training samples with two different interpolating algorithms. Though [12] chooses to use "bilinear" algorithm to resize the videos, we use "area" interpolating algorithm as it gives better visual quality after resizing.

4.2 TRAINING DETAILS

We train our models with a batch size of 108, which is the largest that can fit on a p3.16xlarge instance. Both $\mathcal{D}_T$ and $\mathcal{D}_S$ are updated twice for every $\mathcal{G}$ update, same as [12]. Adam [48] is used to optimize the DVDGAN model. Both the latent vector $\mathbf{z} \sim \mathcal{N}(0, I)$ and the linearly embedded class vector $e(y)$ have 128 dimensions. The learning rate for $\mathcal{G}$ and both $\mathcal{D}$s are 5e$-$4 and 1e$-$4, respectively. Separable attention is only used in $\mathcal{G}$ at resolution 32. Generated videos are 64×64 resolution and 12 frames long. The channel multiplier ch in [12] is 128 for 64×64 videos. We use 64 for generator and both discriminators due to the limited computing resource. The best FID and IS scores are achieved between 100,000 and 110,000 steps. Most models are trained for 96 to 120 hours with a single p3.16xlarge instance. A complete table of hyperparameters and design choices can be find in Table 4.1.

Figure 4.1: 64 × 64 pixel resized UCF101 training samples. **Top three rows**: "area" interpolating algorithm. **Bottom three rows**: "bilinear" interpolating algorithm. It is obvious that "area" mode gives smoother and less pixelated output.

4.3 OVERFITTING

GANs can be hard to train as it is easy to overfit on the training data. The two most common overfitting cases we observe are GAN memorization and mode collapse. We will explain why they exist and what we do to mitigate them.

4.3.1 GAN Memorization

An ideal GAN should generalize well from the training data, which requires the networks to learn meaningful features and produce new samples that are not found in the training data. In practice, GANs have the tendency to simply memorize the training data [6, 12, 49]. In the setup of class-conditional generative task, *GAN memorization* appears as limited diversity of generated samples within each class. The GAN memorizes a small number of samples for

Hyperparameter	Value
Batch size	108
Num. of $\mathcal{D}$ updates per $\mathcal{G}$ update	2
$\mathcal{G}$ learning rate	5e−4
$\mathcal{D}$ learning rate	1e−4
Beta1 for $\mathcal{G}$'s Adam optimizer	0.0
Beta1 for $\mathcal{D}$'s Adam optimizer	0.0
Beta2 for $\mathcal{G}$'s Adam optimizer	0.999
Beta2 for $\mathcal{D}$'s Adam optimizer	0.999
Resolution of the separable attention layer in $\mathcal{G}$	32
Num. of frames sampled (k value) for $\mathcal{D}_S$	8
Activation function for $\mathcal{G}$	ReLU
Activation function for $\mathcal{D}$	ReLU
Initialization style for $\mathcal{G}$	$\mathcal{N}(0, 0.02I)$
Initialization style for $\mathcal{D}$	$\mathcal{N}(0, 0.02I)$
Dimension of latent vector $\mathbf{z}$	128
Dimension of class embedding $e(y)$	128
channel multiplier ch for $\mathcal{G}$	64
channel multiplier ch for $\mathcal{D}$	64

Table 4.1: Hyperparameters and design choices used for 64 × 64, 12-frame DVDGAN model. Anything not included in the table is aligned with default values in the BigGAN model.

each class it has "seen" during the training phase and is not capable of producing "unseen" samples. [12] reports significant GAN memorization for DVDGAN trained on UCF101. Nagarajan et al. [49] suggest stronger levels of GAN memorization can be achieved when:

1. The generator $\mathcal{G}$ is trained on a sufficiently large set of distinctive latent vectors, or
2. The batch size is not large enough.

BigGAN proposes several useful methods to mitigate GAN memorization. To address the issue of the first bullet point, BigGAN applies the "truncation trick" and "early stopping" to cut down the range of the latent space and keep the networks from seeing too many latent vectors, respectively. When it comes to the batch size, BigGAN increases its batch size to as large as 2048. These methods have been proven effective in practice by both BigGAN and DVDGAN. However, because DVDGAN's architecture is more complicated than BigGAN, when it is trained on a relatively small dataset, DVDGAN still has a strong tendency to simply memorize some training samples. To show this phenomenon, we train our DVDGAN on UCF101 and conduct class-wise nearest neighbor analysis on selected generated videos (Figure 4.2).

Figure 4.2: Right: Generated samples. Left: Class-wise nearest neighbors (closest clip group) in pixel-space from UCF101. Class (from top to bottom): Playing Flute, Wall Pushups, Jump Rope, Tennis Swing, Playing Guitar and Playing Tabla.

Because of the lack of complexity and diversity, UCF101 is not suitable for large and complex models like DVDGAN. We will show later that Kinetics-400 with a size nearly 40 times larger than UCF101 can greatly reduce the level of GAN memorization.

4.3.2 Mode Collapse

The objective functions of GANs does not motivate them to learn diverse samples because the discriminator only checks if the samples are fake or not, but not if they are different from each other. Therefore, the generator often finds a way to "cheat" by producing same or very similar samples for the same class. This failure case is called *mode collapse* [50]. Under the class-conditional GAN setup, embedded class vector is concatenated with noise vector. Noise vectors are often neglected because it is a lot easier for conditional GANs to learn conditional contexts from class embeddings because they are more structured [51]. When trained on a small and simple dataset, mode collapse can be a major issue for GAN. In Figure 4.3, We show an example of mode collapse of DVDGAN trained on UCF101.

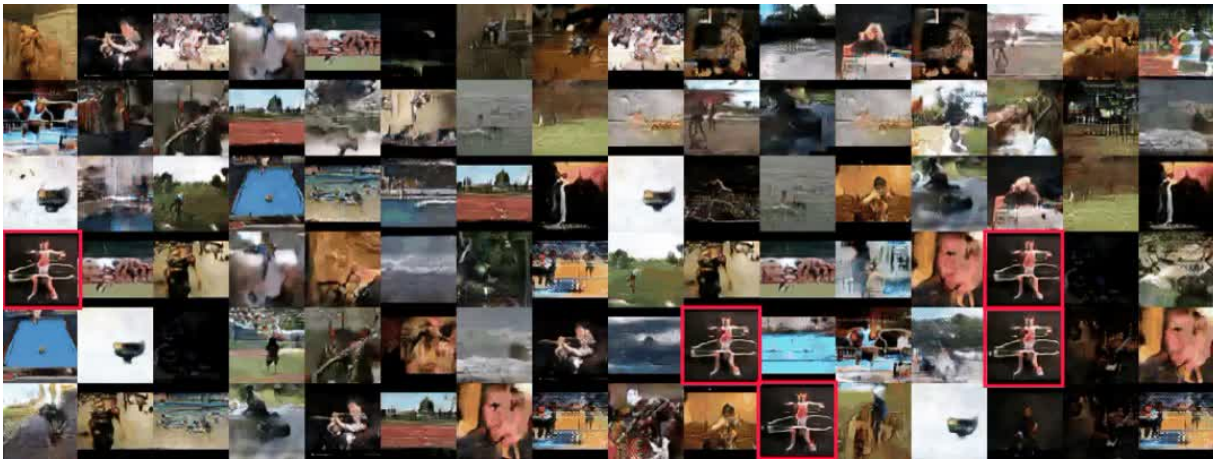


Figure 4.3: An example of mode collapse. This is a batch of generated videos from DVDGAN trained on UCF101 at 43200 step. The videos in the red boxes have the same class label "Hula Hoop". The generator can only generate one mode for this specific class.

4.4 EVALUATION METRICS

Despite substantial progress in terms of resolution and visual quality, how to evaluate and compare GANs remains to be a task that is far from being tackled. A variety of quantitative GAN evaluation measures have been proposed. Among them, two metrics that require a pre-trained neural network, Inception Score (IS) and Fréchet Inception Distance (FID), are

perhaps the most popular ones. IS measures the expectation of the KL divergence between the conditional label distribution $p(y|\mathbf{x})$ and its marginal distribution $p(y)$ [43, 52]. GANs with stronger capacity to produce samples that are both diverse and highly classifiable will create a larger discrepancy between the two distributions, resulting in higher IS score. Experiments show IS has good correlation with human judgement [43]. Despite its popularity, IS has a hard time detecting GAN memorization, overfitting and mode collapse [53, 54]. FID, introduced in 2017 [44], is considered to be a more robust alternative than IS [52]. FID assumes embedded synthetic and real samples form two continuous multivariate Gaussian distributions. By measuring the distance between synthetic and real data distributions, FID quantifies how similar are the synthetic samples to the real data. As opposed to IS, FID score is lower when the quality of generated data is better [44]. FID can detect intra-class mode collapse, which will fail IS [52].

Recently, a lot of alternative GAN evaluation metrics are introduced with advantages in different aspects, including Mode Score (MS) [55], low-level image statistics [56], Geometry Score [57], number of statistically-different bins (NDB) [58], etc. In this work, we report IS and FID as they are still the most common evaluation metrics for GAN. The original IS and FID use the Inception network [59] trained on the ImageNet dataset [23] to extract features. In order to measure video data, Inception network is replaced with publicly available[1] ResNet (2+1)D network [60] pre-trained on Kinetics-400 [13]. This makes our FID really similar to the Fréchet Video Distance (FVD) proposed in [61].

4.5 COLLAPSE TO DARK VIDEOS

During our training, we observe a failure case that has not been documented before. The proportion of black/dark videos increases as the training time grows. This failure case is consistent across different training datasets and architecture choices. Figure 4.4 shows the proportion of black videos in a batch of generated samples at different stages of training. Figure 4.4 also implies that the drop of pixel intensity value does not happen to all videos evenly. When the majority of the generated samples become completely or nearly completely black, there are still samples that are very bright. To better understand this failure case, we also plot how average pixel values of the three color channels change over time. According to Figure 4.5, all three channels behave similarly as the number of training step grows: mean pixel value decreases at a slow but steady rate until around 70,000 steps; after 70,000 steps, mean pixel value starts to decrease rapidly. The plots of FID and IS (Figure 4.6)

[1]Further information about the model can be found at `https://pytorch.org/docs/stable/torchvision/models.html#resnet-2-1-d`

quantitatively show how decreasing pixel value influence the quality of the generated samples. Both FID and IS quickly deteriorate after 70,000 steps, which makes it impossible to train the model any longer without getting worse output quality.

To address this problem, we first propose an additional loss to take the change of pixel intensity into account when updating for $\mathcal{G}$. The Average Pixel Value Loss $\mathcal{L}_p$ is defined as:

$$\mathcal{L}_p = \alpha \frac{1}{n} \sum_{t,h,w,c} \|x_{t,h,w,c} - \mathcal{G}(z)_{t,h,w,c}\|, \tag{4.1}$$

where α is the weight, n is the dimension of the samples, t is the number of frames per sample, h is the height of the sample, w is the width of the sample and c is the number of channels. $\mathcal{L}_p$ is added to the original DVDGAN loss function for $\mathcal{G}$.

We report experimental results for four models trained on Kinetics-400 dataset from scratch with different weights and starting points. Because the fast decline in pixel intensity does not happen until a relatively late stage, we let the Average Pixel Value loss to kick in after 50,000 steps for all but one trials. One trial will have the Average Pixel Value Loss from the beginning of its training. We also experiment on different weight factors α. The experimental results for the four trials can be found in Figure 4.7. Comparing the blue, the orange and the green lines, we find that the Average Pixel Value Loss can decrease the slope of the declining mean pixel value curve and push back the "turning point". In addition, larger weights make more significant improvement. The line of $\alpha = 2$ does not reach the turning point until 100,000 steps while the line of $\alpha = 1$ starts to rapidly drop at step 70,000. However, having the loss at an early stage can make the training results even worse. The red line, which applies the loss from the beginning of the training, starts to crash at step 40,000. When the Average Pixel Value Loss kicks in at step 50,000 for the green line, the red line is already nearly 0.2 lower than the green line. At 80,000 steps, the generated videos of the red model are nearly all black. Figure 4.8 also confirms the above observations. However, the plots of FID and IS (Figure 4.9) show that even though the Average Pixel Value Loss helps to delay the turning point, it cannot eliminate it. FID and IS curves are unstable and have strong tendency to quickly bounce back and deteriorate after reaching their best results. Therefore, we decide to not include the Average Pixel Value Loss in our final implementation.

Loss functions are also potential cause of the black videos. According to [12], the complete objective functions for the two discriminators $\mathcal{D}_S$ and $\mathcal{D}_T$ and the generator $\mathcal{G}$ in practice are:

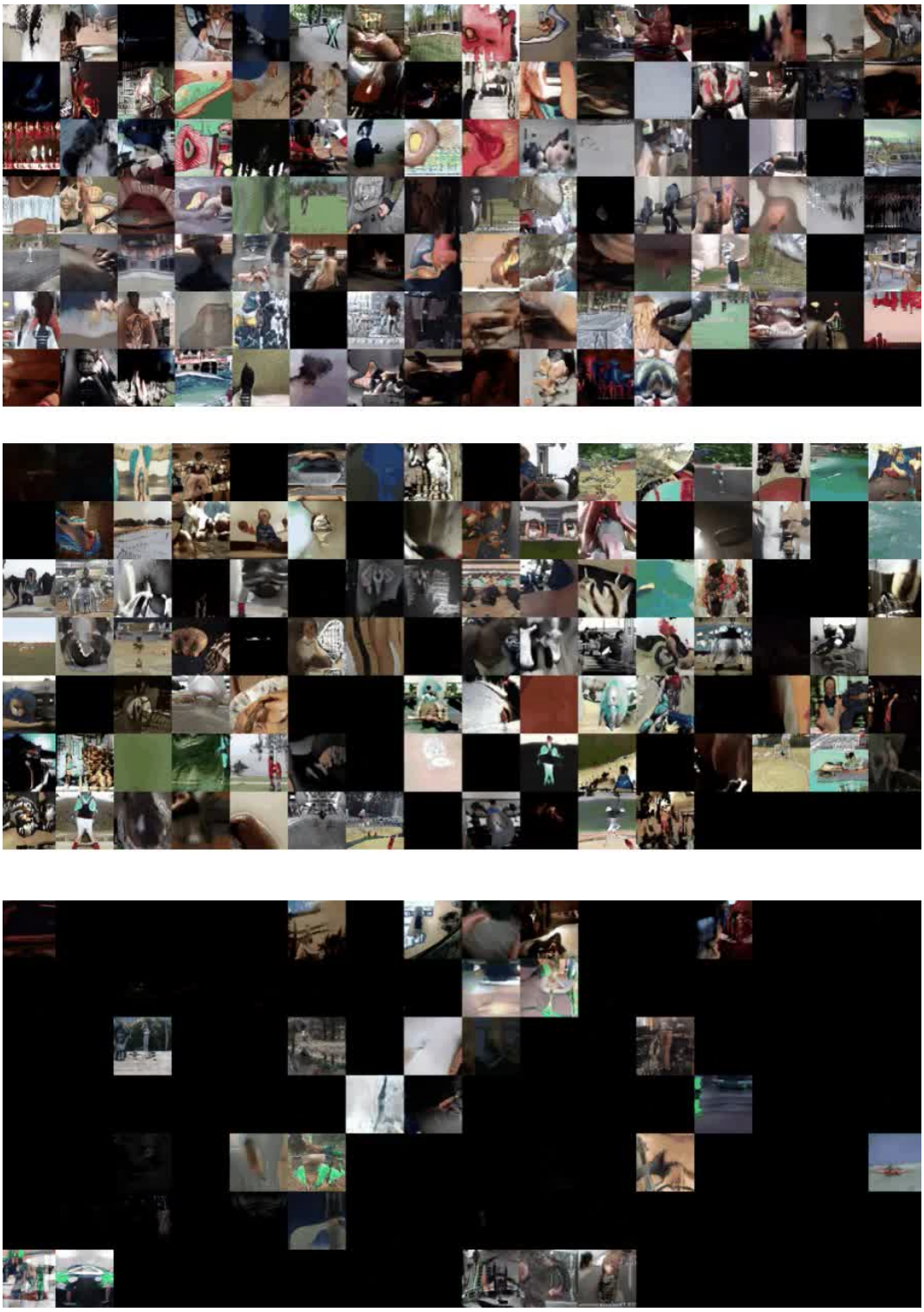


Figure 4.4: Increasing number of black videos at different stages of training. From top to bottom are batches of generated videos at step 54200, 75000 and 106900.

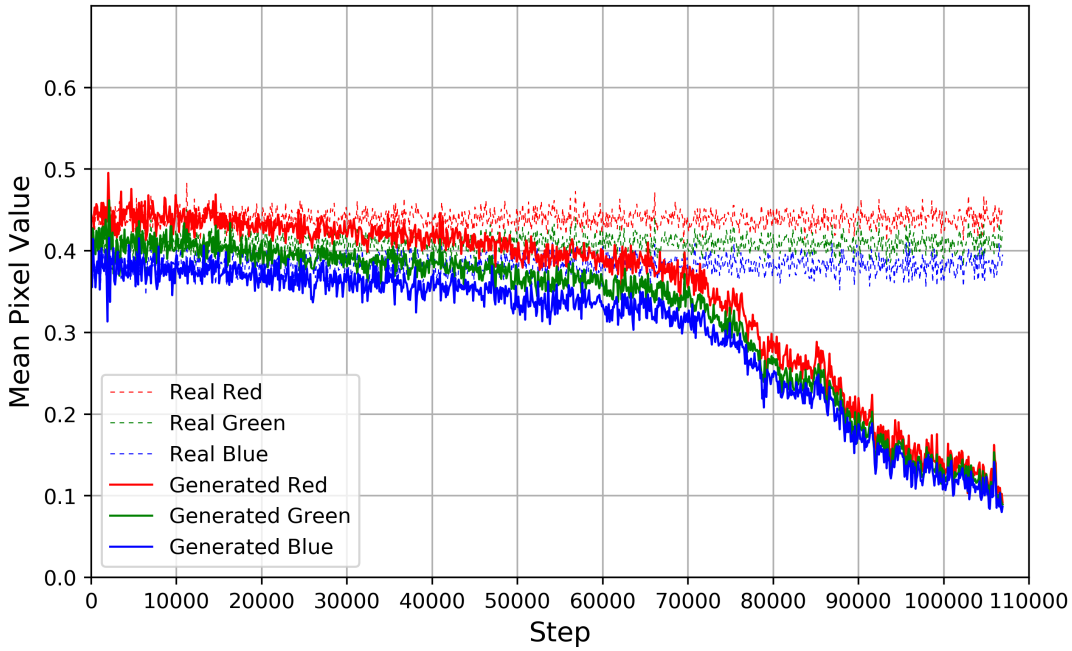


Figure 4.5: A typical plot of mean pixel value of the three color channels vs. training step. Each data point is the average pixel value of an entire batch of generated samples. Different colors represent different color channels. Dotted lines represent real samples and solid lines represent samples generated by $\mathcal{G}$. The range of the pixel value is $[0, 1]$.

$$
\begin{aligned}
\mathcal{D} : \min_{\mathcal{D}} \mathbb{E}_{\mathbf{x}\sim P_{data}(\mathbf{x})} \Big[\min\{0, 1 - \sum_k \mathcal{D}_S(x_k)\} + \min\{0, 1 - \mathcal{D}_T(x)\} \Big] \\
+ \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})} \Big[\min\{0, 1 + \sum_k \mathcal{D}_S(\mathcal{G}(z)_k)\} + \min\{0, 1 + \mathcal{D}_T(\mathcal{G}(z))\} \Big], \\
\mathcal{G} : \max_{\mathcal{G}} \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})} \Big[\sum_k \mathcal{D}_S(\mathcal{G}(z)_k) + \mathcal{D}_T(\mathcal{G}(z)) \Big],
\end{aligned}
\tag{4.2}
$$

where k is the randomly sampled full-resolution frame from the generated video. We follow the hyperparameter choice in [12] and randomly sample 8 out of 12 frames from each video for $\mathcal{D}_S$ to critique individually. We suspect summing the per-frame scores before doing the ReLU operation could result in decrease in pixel value. We improve the objective functions by applying ReLU on individual per-frame scores output by $\mathcal{D}_S$ *prior* to the summation. Adding up scores from $\mathcal{D}_S$ will create an imbalance of scale between the loss of the spatial discriminator and the loss of the temporal discriminator. As a result, the model will learn

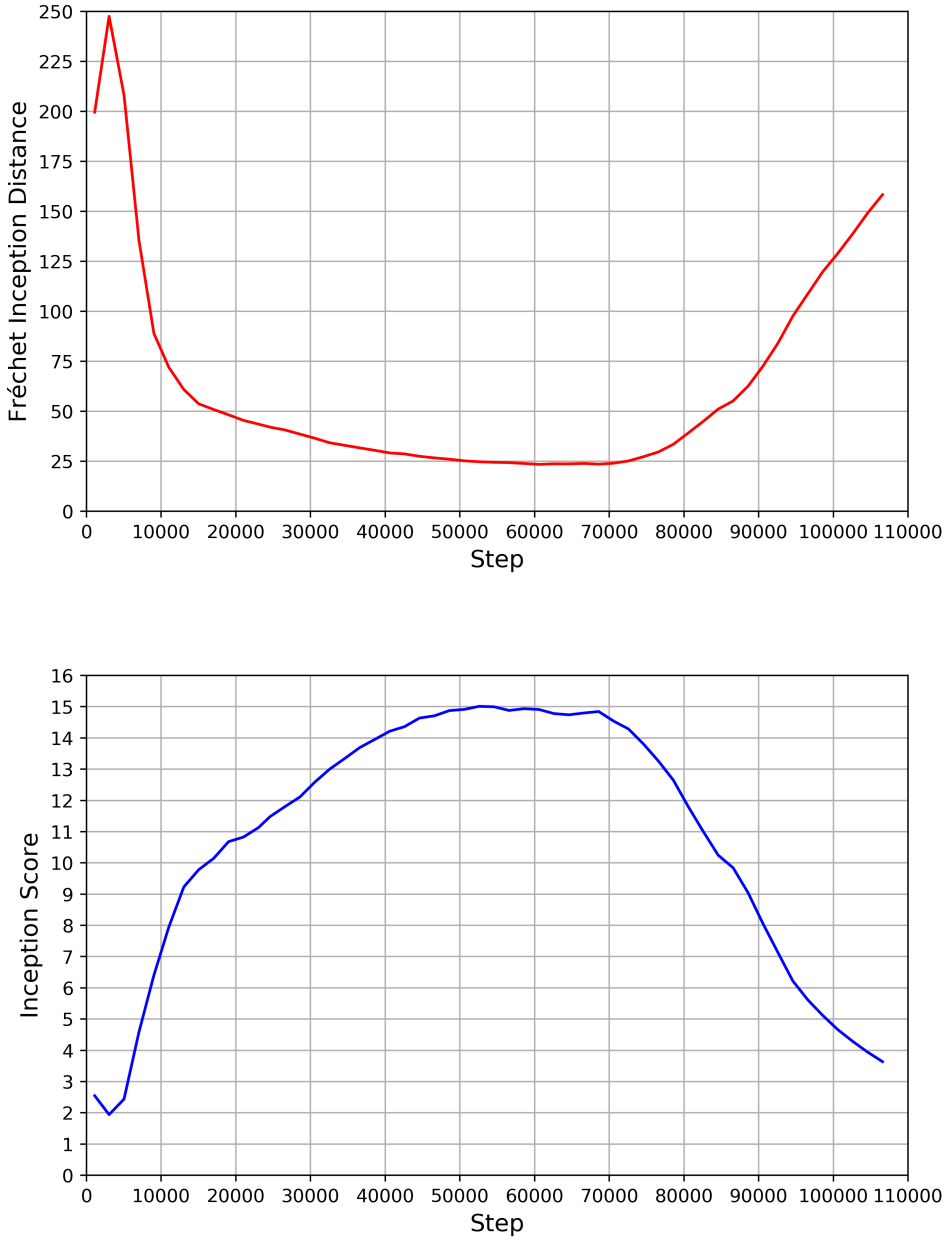


Figure 4.6: Plots of FID (top) and IS (bottom) for the same training as in Figure 4.5. Both metrics keep getting better for the first 50,000 steps and plateau between 50,000 and 70,000. The rapid drop of pixel intensity after 70,000 steps results in a quick deterioration for both metrics after 70,000.

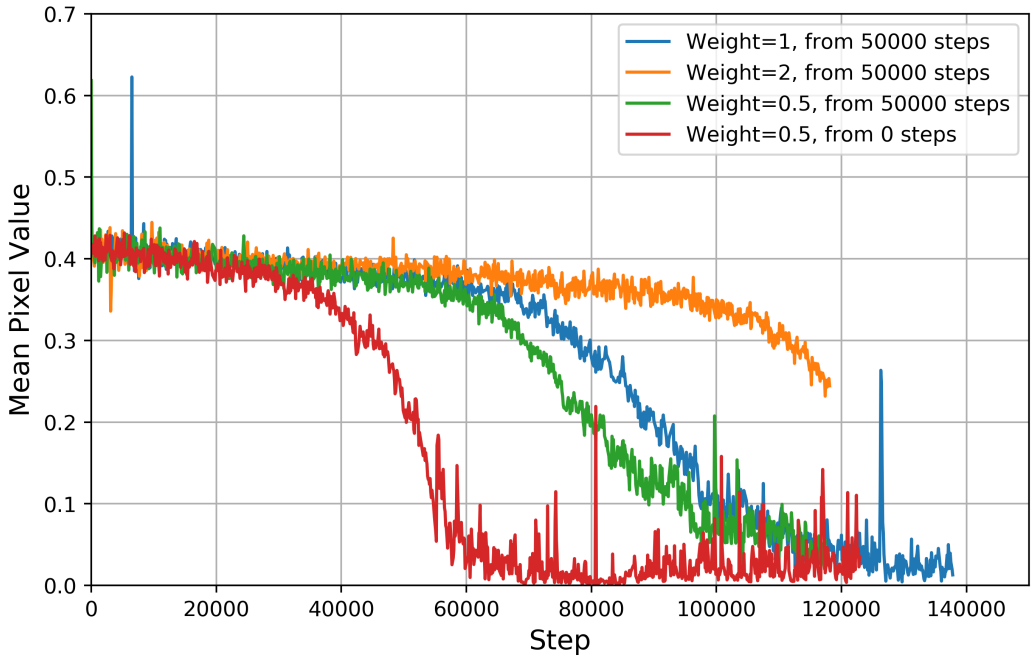


Figure 4.7: Comparison of the mean pixel values for models trained with the Average Pixel Value Loss. Each data point is the mean pixel value across all three color channels for an entire batch of generated samples.

to ignore temporal dynamics and produce samples with almost no movement. We solve the problem by taking the average of the $\mathcal{D}_S$ scores rather than the summation. Formally, the new objective functions are given by:

$$\begin{aligned}\mathcal{D}: \min_{\mathcal{D}} \mathbb{E}_{\mathbf{x}\sim P_{data}(\mathbf{x})}\Big[\frac{1}{K}\sum_k \min\{0, 1-\mathcal{D}_S(x_k)\} + \min\{1-\mathcal{D}_T(x)\}\Big] \\ + \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})}\Big[\frac{1}{K}\sum_k \min\{0, 1+\mathcal{D}_S(\mathcal{G}(z)_k)\} + \min\{0, 1+\mathcal{D}_T(\mathcal{G}(z))\}\Big], \\ \mathcal{G}: \max_{\mathcal{G}} \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})}\Big[\frac{1}{K}\sum_k \mathcal{D}_S(\mathcal{G}(z)_k) + \mathcal{D}_T(\mathcal{G}(z))\Big].\end{aligned} \tag{4.3}$$

We show in Figure 4.10 and Figure 4.11 that this modification to the objective functions, even though still does not solve the black video problem entirely, achieves similar or even better results than the Average Pixel Value Loss with $\alpha = 2$ does.

To further narrow down the problem, an ablation study is conducted. We remove certain modules from the model, one at a time, and train the model with everything else unchanged

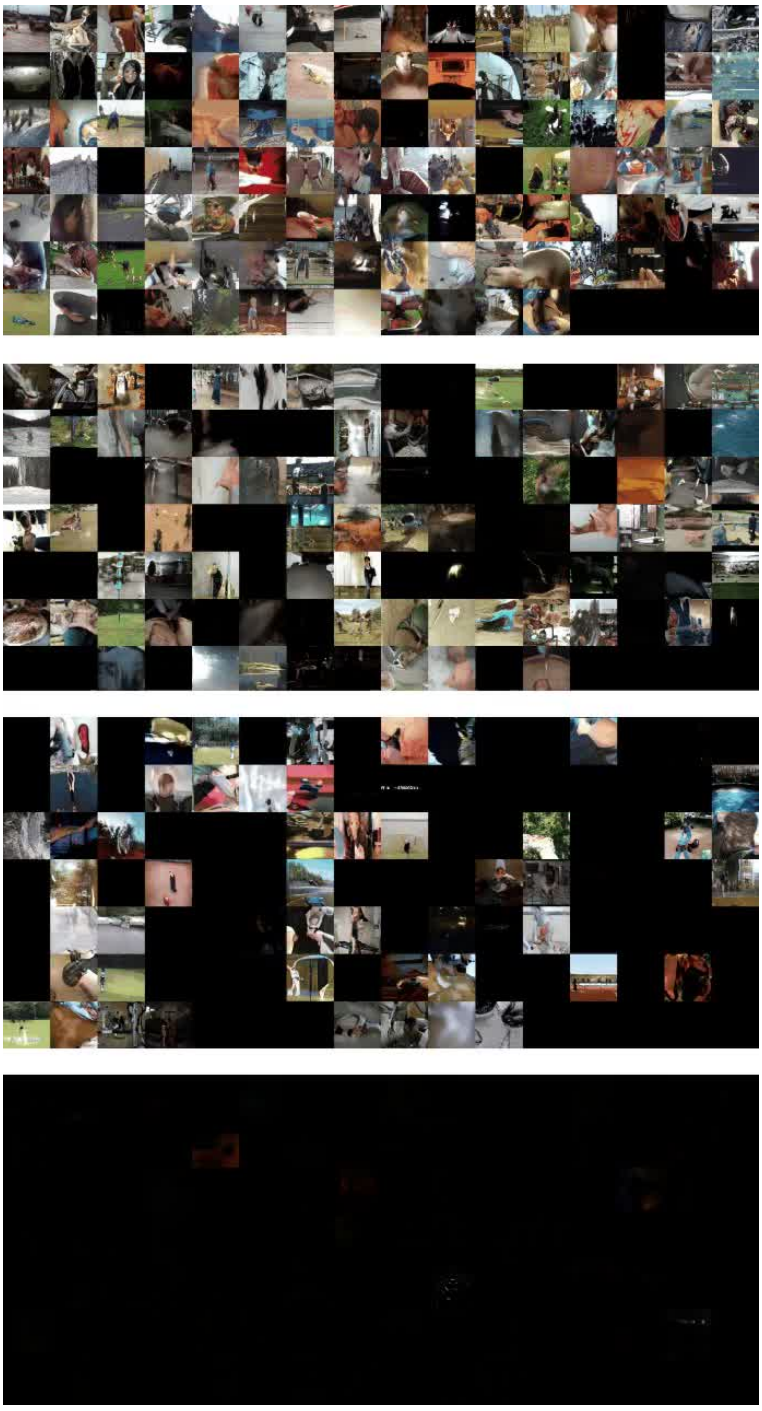


Figure 4.8: Batches of generated samples from models trained with the Average Pixel Value Loss. All batches are generated at step 83,300. From top to bottom, the weight and starting point of the Average Pixel Value Loss are set to (2, 50,000), (1, 50,000), (0.5,50,000) and (0.5, 0).

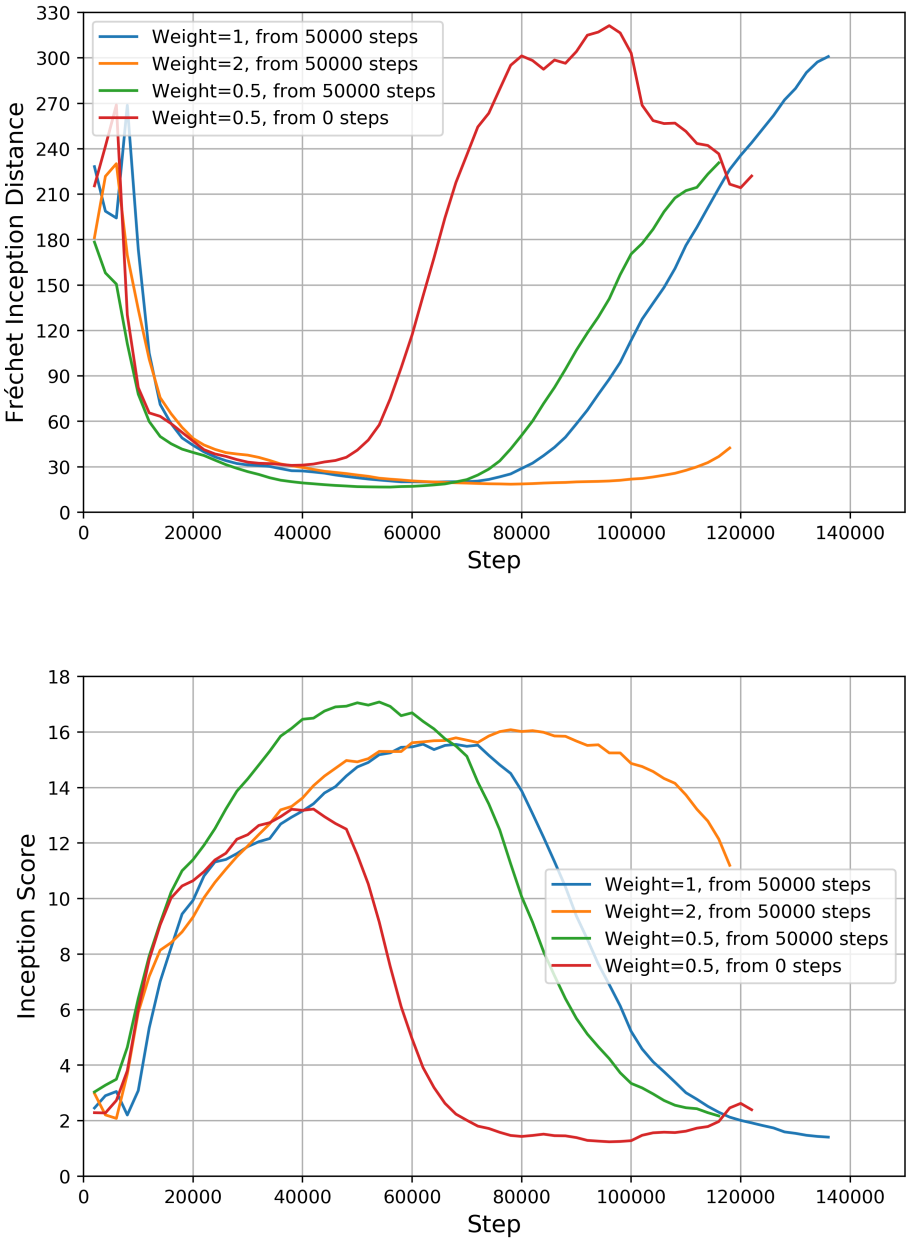


Figure 4.9: Plots of FID (top) and IS (bottom) for the four experiments as in Figure 4.7.

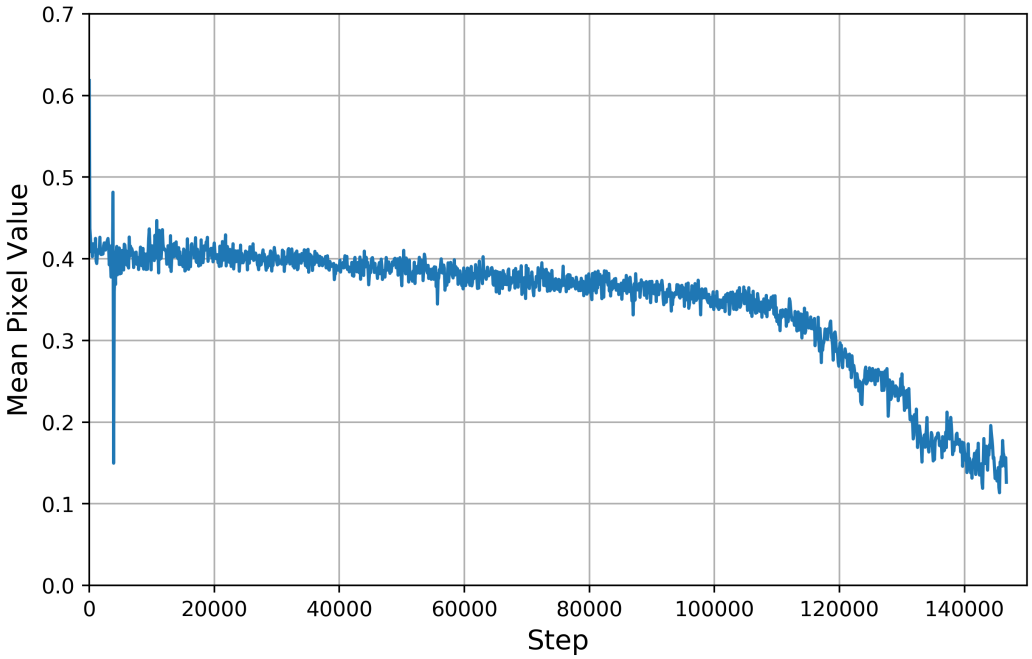


Figure 4.10: The mean pixel values of the model trained with the new objective functions. Each data point is the mean pixel value across all three color channels for an entire batch of generated samples.

to examine the mean pixel value curve. We find decreasing sample pixel value is directly connected to the presence of $\mathcal{D}_T$. Figure 4.12 shows that when $\mathcal{D}_T$ is turned off, the curve of mean pixel value is almost flat after the initial fluctuation. Therefore, it is safe to draw the conclusion that certain design choices for $\mathcal{D}_T$ cause the "black video" failure case.

$\mathcal{D}_T$ contains two 3-D residual blocks followed by two 2-D residual blocks. Originally the time dimension of the tensor that comes out of the last 3-D block is folded into the channel dimension before forward passing to the 2-D blocks. We modify the architecture of $\mathcal{D}_T$ so that the time dimension is now folded into the batch dimension. Keeping the channel dimension consistent across the entire temporal discriminator helps to stabilize the training and mitigate the "black video" problem. In Figure 4.13 we show that by applying the new objective functions and modifying the temporal discriminator $\mathcal{D}_T$'s structure, we successfully eliminate the "turning point" in the mean pixel value curve. Even though the mean pixel value still decreases at a very slow rate over time, it no longer negatively effects the evaluation metrics, which is confirmed by Figure 4.14.

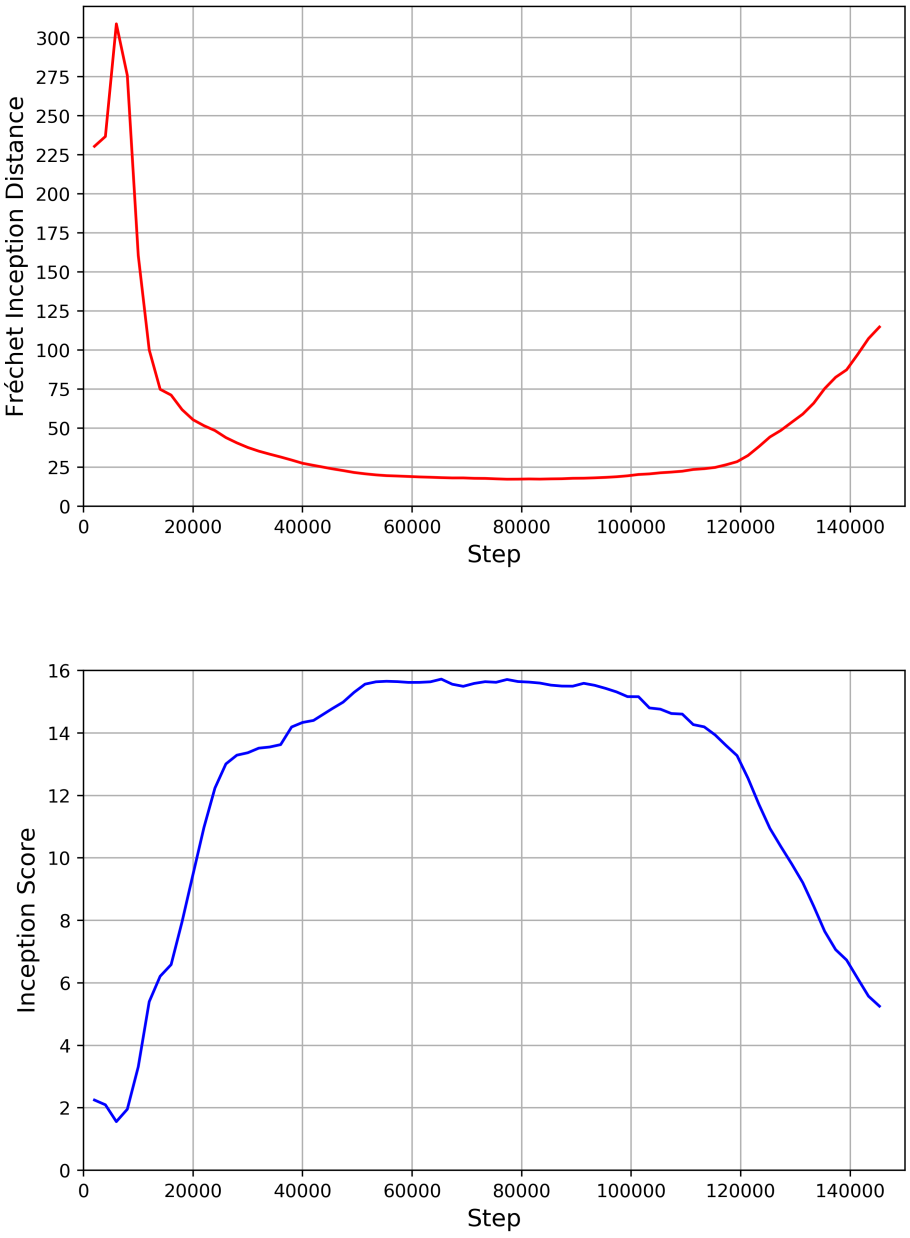


Figure 4.11: FID (top) and IS (bottom) for the same experiment as in Figure 4.10.

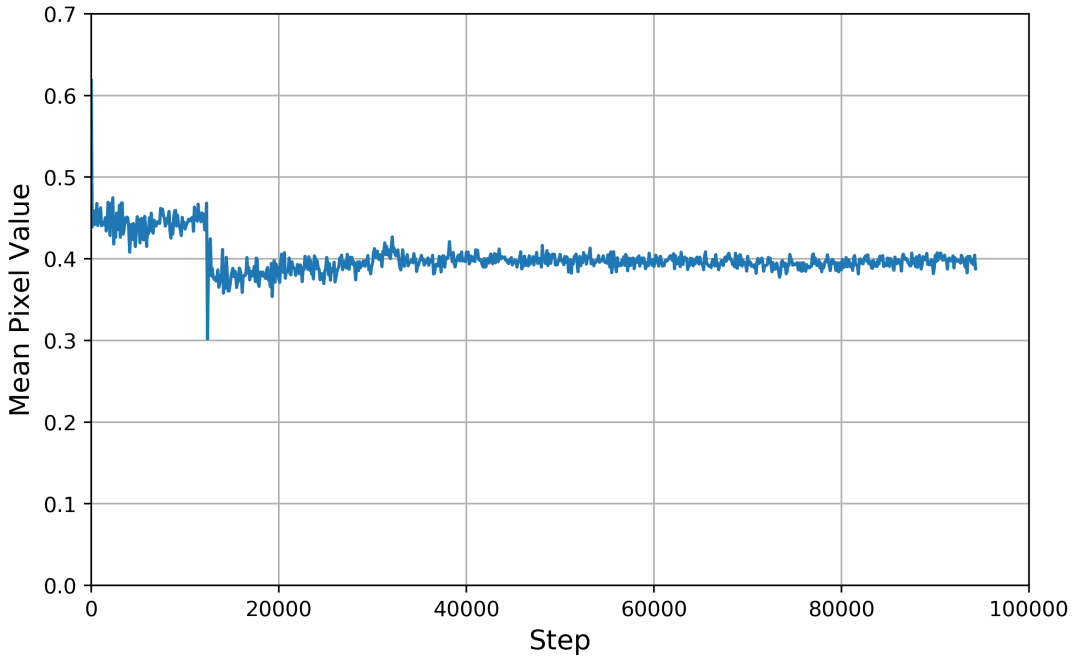


Figure 4.12: The mean pixel values of the model trained without $\mathcal{D}_T$. Each data point is the mean pixel value across all three color channels for an entire batch of generated samples.

4.6 MISCELLANEOUS EXPERIMENTS

Below we list some miscellaneous experiments we conducted and their results.

- We replace the two 3-D residual blocks in $\mathcal{D}_T$ with two standard 3-D convolutional layers and there are no big differences in visual quality. We suggest to use the standard 3-D convolutional layer for low-resolution video generation as it reduces computational costs.

- Although [12] uses a learning rate of 1e−4 and 5e−4 for $\mathcal{G}$ and $\mathcal{D}$ respectively, our experiments show a higher learning rate for the generator produces better results. The results we report are all trained with a learning rate of 5e−4 and 1e−4 for $\mathcal{G}$ and $\mathcal{D}$ respectively.

4.7 FINAL RESULTS

We train our final version of the DVDGAN model with an early stopping at 110,000 steps. Our model trained on Kinetics-400 64 $\times$ 64, 12 frames videos is able to achieve 11.92 and

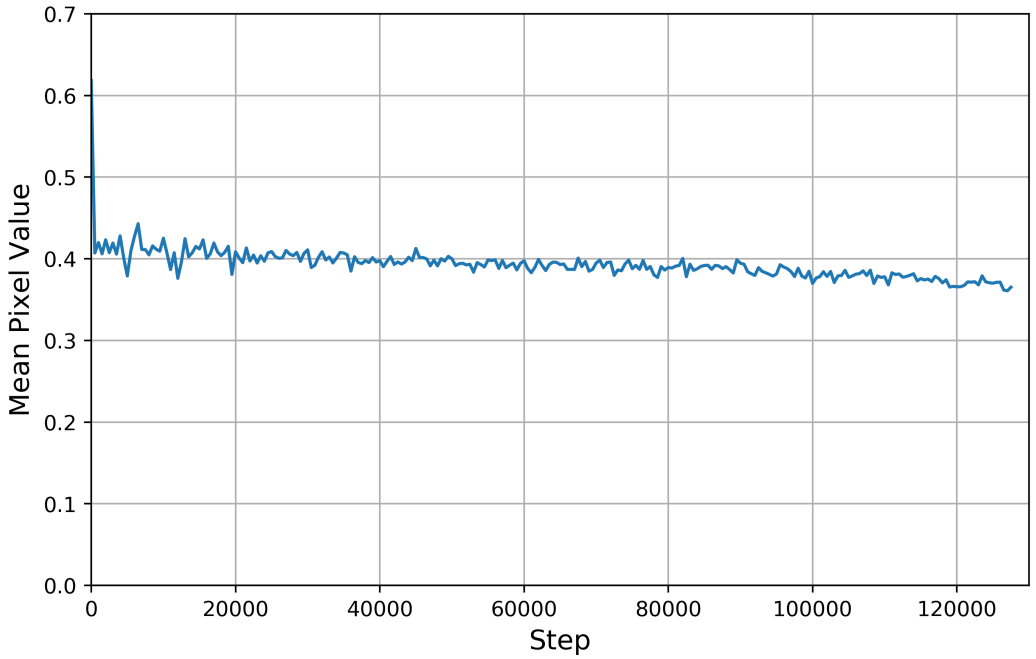


Figure 4.13: The mean pixel values of the model trained with both the new objective functions and modified $\mathcal{D}_T$. Each data point is the mean pixel value across all three color channels for an entire batch of generated samples.

20.76 for FID and IS, respectively. A random batch of output videos can be found in Figure 4.15. We also include selected samples from different classes in Figure 4.16.

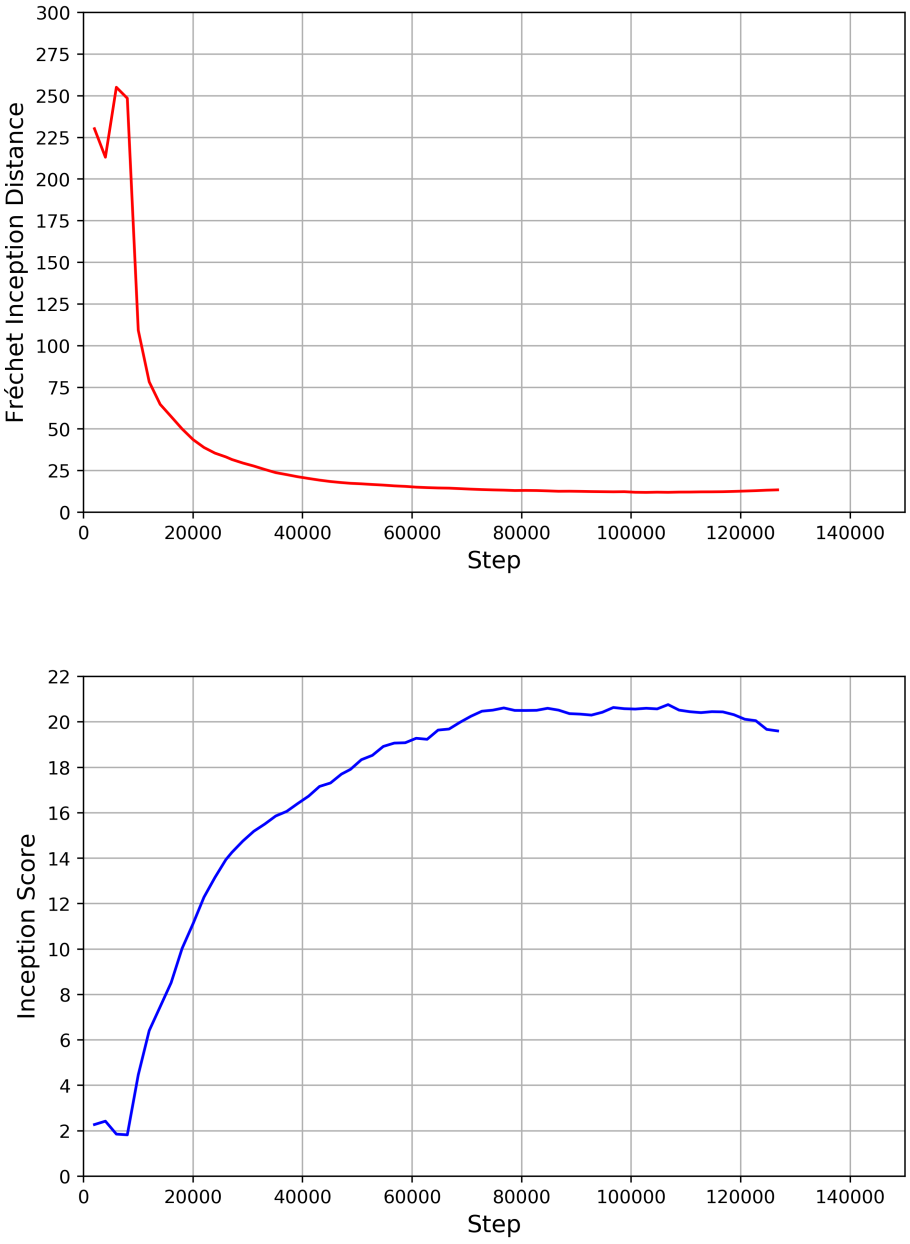


Figure 4.14: FID (top) and IS (bottom) for the same experiment as in Figure 4.13.

Figure 4.15: A random Batch of generated samples from our final model. The output samples are 64 $\times$ 64 resolution and 12 frames long.

(a) Class: *Country line dancing*

(b) Class: *Snowboarding*

(c) Class: *Gymnastics tumbling*

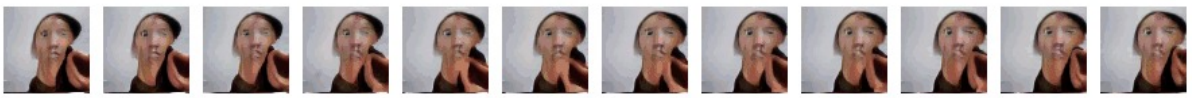
(d) Class: *Frying vegetables*

(e) Class: *Parasailing*

(f) Class: *Making snowman*

(g) Class: *Fixing hair*

(h) Class: *Tasting food*



(i) Class: *Applying cream*

Figure 4.16: Selected output video samples from different classes. Each row is a separate sample.

CHAPTER 5: CONCLUSION

In this work we provided a detailed guideline for training large-scale class-conditional video Generative Adversarial Networks with limited computing resource. We analyzed practical problems including data preprocessing and training details, which are often not given enough attentions and instructions in previous literature. Detailed guidelines were given for different steps of training. We compared different methods for preprocessing and explained the advantages and disadvantages of each method. We also showed what GAN memorization and mode collapse look like in practice for video samples and how more complex and diverse datasets help to solve such problems. One of the most important contributions of this work is documenting the failure case of "black videos", which has not been mentioned by other literature before. Extensive analysis and experiments were carried out to identify and mitigate this problem. We hope researchers who are interested in training conditional video GANs find this work useful as a guideline.

We provided some advice in this work on reducing computing resources needed for training video GANs. In the future we plan to investigate methods that further reduce computational costs of training large-scale video GANs. Liu *et al.* propose a two-stage acceleration framework for high resolution image synthesis using shorter training time and less computational resources [62]. The framework generates small 2-D latent codes and uses pre-trained decoder to transform the codes to full resolution image samples. Integrated with convolutional GRU, we hope the framework can greatly reduce the training time of DVDGAN. *Transfer learning* is another potential way to accelerate the training process. Frégier *et al.* [63] and Wang *et al.* [64] study the application of transfer learning in the context of image GANs. They show that leveraging knowledge from pre-trained GANs helps the model to converge faster. Whether the same method is effective for generative models of natural video remains unknown. We leave this task as a future research direction.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[2] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE, 2017, pp. 1–7.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[4] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[5] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[6] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[7] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," *arXiv preprint arXiv:1708.05509*, 2017.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[9] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances in neural information processing systems*, 2016, pp. 613–621.

[10] M. Saito, E. Matsumoto, and S. Saito, "Temporal generative adversarial nets with singular value clipping," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2830–2839.

[11] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1526–1535.

[12] A. Clark, J. Donahue, and K. Simonyan, "Adversarial video generation on complex datasets," *arXiv preprint arXiv:1907.06571*, 2019.

[13] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev et al., "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[14] D. Acharya, Z. Huang, D. P. Paudel, and L. Van Gool, "Towards high resolution video generation with progressive growing of sliced wasserstein gans," *arXiv preprint arXiv:1810.02419*, 2018.

[15] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): A survey," *IEEE Access*, vol. 7, pp. 36 322–36 333, 2019.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[18] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[20] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[22] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[24] M. Saito and S. Saito, "Tganv2: Efficient training of large models for video generation with multiple subsampling layers," *arXiv preprint arXiv:1811.09245*, 2018.

[25] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, "Video-to-video synthesis," *arXiv preprint arXiv:1808.06601*, 2018.

[26] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, "Everybody dance now," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5933–5942.

[27] Y. Zhou, Z. Wang, C. Fang, T. Bui, and T. Berg, "Dance dance generation: Motion transfer for internet videos," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[28] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, "Recycle-gan: Unsupervised video retargeting," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 119–135.

[29] Y. Wu, V. Singh, and A. Kapoor, "From image to video face inpainting: Spatial-temporal nested gan (stn-gan) for usability recovery," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 2396–2405.

[30] C. Wang, H. Huang, X. Han, and J. Wang, "Video inpainting by jointly learning temporal structure and spatial details," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5232–5239.

[31] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[32] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," *arXiv preprint arXiv:1606.01933*, 2016.

[33] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[34] J. H. Lim and J. C. Ye, "Geometric gan," *arXiv preprint arXiv:1705.02894*, 2017.

[35] D. Tran, R. Ranganath, and D. M. Blei, "Deep and hierarchical implicit models," *arXiv preprint arXiv:1702.08896*, vol. 7, p. 3, 2017.

[36] T. Miyato and M. Koyama, "cgans with projection discriminator," *arXiv preprint arXiv:1802.05637*, 2018.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[38] N. Aifanti, C. Papachristou, and A. Delopoulos, "The mug facial expression database," in *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*. IEEE, 2010, pp. 1–4.

[39] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2. IEEE, 2005, pp. 1395–1402.

[40] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[41] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, 2015, pp. 843–852.

[42] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," *arXiv preprint arXiv:1808.01340*, 2018.

[43] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[44] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, 2017, pp. 6626–6637.

[45] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," *arXiv preprint arXiv:1511.06432*, 2015.

[46] "Amazon web services (aws) - cloud computing services," 2020. [Online]. Available: aws.amazon.com

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[49] V. Nagarajan, C. Raffel, and I. J. Goodfellow, "Theoretical insights into memorization in gans."

[50] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[51] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1429–1437.

[52] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, 2019.

[53] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 2018.

[54] J. Yang, A. Kannan, D. Batra, and D. Parikh, "Lr-gan: Layered recursive generative adversarial networks for image generation," *arXiv preprint arXiv:1703.01560*, 2017.

[55] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016.

[56] Y. Zeng, H. Lu, and A. Borji, "Statistics of deep generated images," *arXiv preprint arXiv:1708.02688*, 2017.

[57] V. Khrulkov and I. Oseledets, "Geometry score: A method for comparing generative adversarial networks," *arXiv preprint arXiv:1802.02664*, 2018.

[58] E. Richardson and Y. Weiss, "On gans and gmms," in *Advances in Neural Information Processing Systems*, 2018, pp. 5847–5858.

[59] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[60] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[61] T. Unterthiner, S. van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly, "Towards accurate generative models of video: A new metric & challenges," *arXiv preprint arXiv:1812.01717*, 2018.

[62] J. Liu, Y. Yao, and J. Ren, "An acceleration framework for high resolution image synthesis," *arXiv preprint arXiv:1909.03611*, 2019.

[63] Y. Frégier and J.-B. Gouray, "Mind2mind: transfer learning for gans," *arXiv preprint arXiv:1906.11613*, 2019.

[64] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu, "Transferring gans: generating images from limited data," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 218–234.